CT437 COMPUTER SECURITY AND FORENSIC COMPUTING

DEFINITIONS, TERMINOLOGY, AND CASE STUDIES

Dr. Michael Schukat

Ollscoil na Gaillimhe
University of Galway

What is Cybersecurity?

- Computer security, cybersecurity or information technology security (IT security) is the protection of computer systems and networks from the theft of or damage to their hardware, software, or electronic **data**, as well as from the **disruption or misdirection of the services they provide** (Wikipedia), i.e.:
- Protection from cybercrime of
  - data (from theft or manipulation)
  - services (from disruption or misuse)
- This protection can be on a personal, organisational or government level

States of Data

- Data at rest
  - Rest refers to data stored in memory or on a permanent storage device such as a hard drive, solid-state drive or USB drive
- Data in process
  - Processing refers to data that is being used to perform an operation such as updating a database record
- Data in transit
  - Transmission refers to data traveling between information systems, e.g. data transfer over a network via TCP/IP

How to provide Protection?

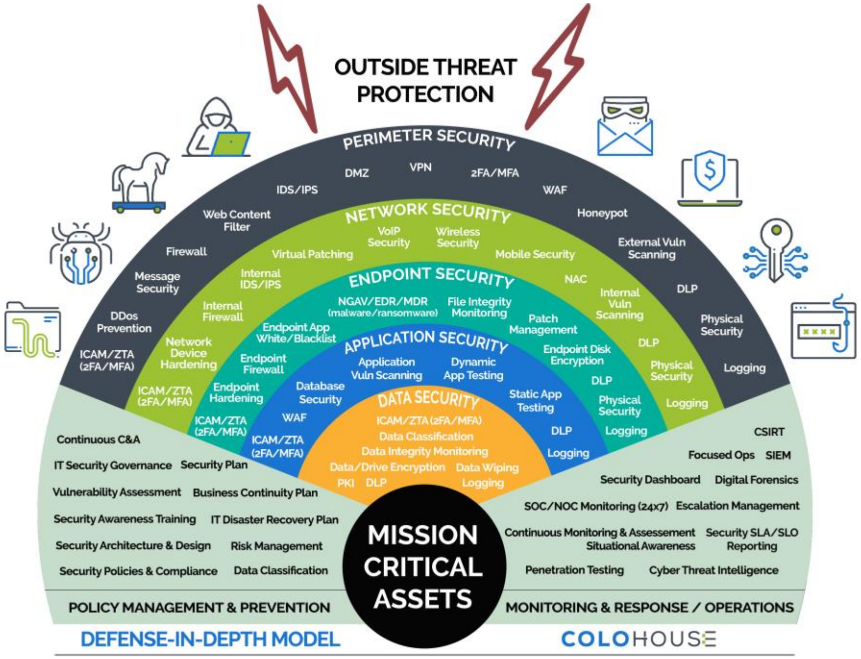
- **Awareness, training and education** are the measures put in place by an organisation to ensure that users are knowledgeable about potential security threats and the actions they can take to protect information systems

- **Technology** refers to the software and hardware-based solutions designed to protect information systems such as firewalls, which continuously monitor your network in search of possible malicious incidents

- **Policy and procedure** refers to the administrative controls that provide a foundation for how an organization implements information assurance, such as incident response plans and best practice guidelines

Defense in Depth

- □ Defense in Depth (DiD) is an approach to cybersecurity in which a series of defensive mechanisms are layered in order to protect assets
- □ If one mechanism fails, another one steps up immediately to thwart an attack

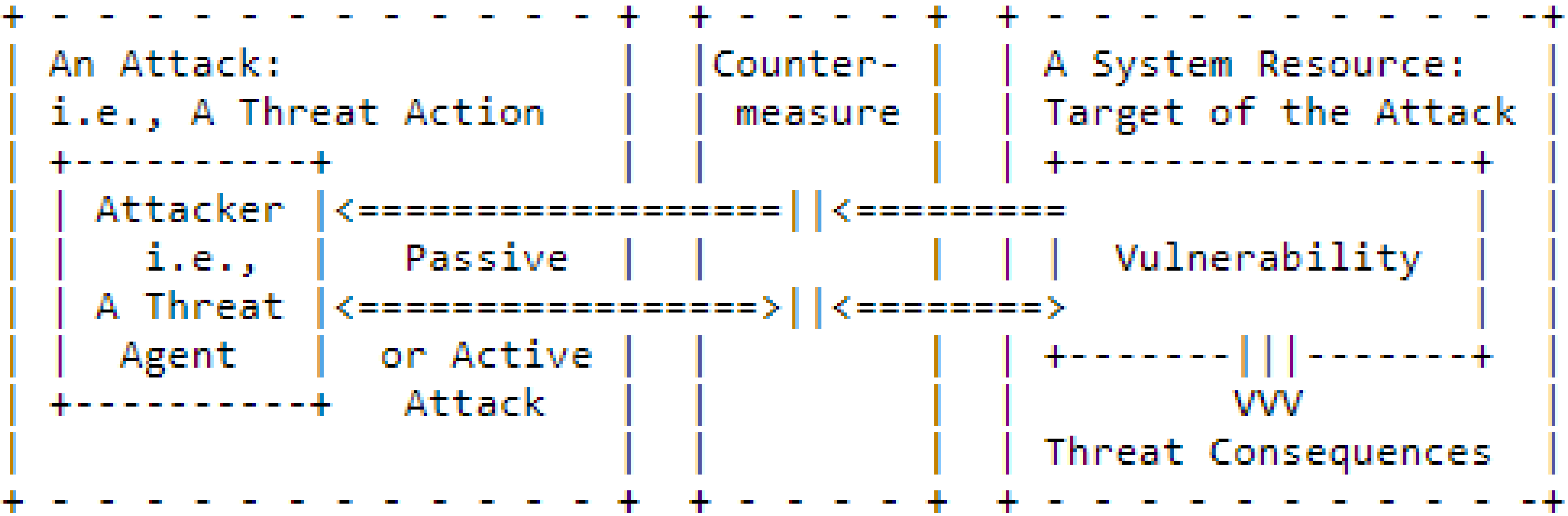


European Union Agency for Cybersecurity (ENISA)

- ENISA is the Union's agency dedicated to achieving a high common level of cybersecurity across Europe
- https://www.enisa.europa.eu/
- ENISA threat landscape report: https://www.enisa.europa.eu/topics/cyber-threats/threat-landscape
- ENISA has also issued a 2024 report *providing policy makers at EU level with an evidence-based overview of the state of play of the cybersecurity landscape and capabilities at the EU, national and societal levels, as well as with policy recommendations to address identified shortcomings and increase the level of cybersecurity across the Union current threat landscape* (see also Canvas)

The big Picture – RFC2828

- RFC2828, Internet Security Glossary
- https://tools.ietf.org/html/rfc2828



```
+ - - - - - - - - - - - - +  + - - - - +  + - - - - - - - - - - -+
| An Attack:              |  |Counter- |  | A System Resource:   |
| i.e., A Threat Action   |  | measure |  | Target of the Attack |
| +----------+            |  |         |  | +-----------------+  |
| | Attacker |<==================||<=========                 |  |
| |   i.e.,  |   Passive  |  |         |  | |  Vulnerability  |  |
| | A Threat |<=================>||<========>                 |  |
| |  Agent   |  or Active |  |         |  | +-------|||-------+  |
| +----------+   Attack   |  |         |  |         VVV          |
|                         |  |         |  | Threat Consequences  |
+ - - - - - - - - - - - - +  + - - - - +  + - - - - - - - - - - -+
```

What is a Threat Agent/Actor?

- The term *threat agent or threat actor* is used to indicate an individual, thing or a group that can manifest a threat
  - In computer security, a threat is a potential negative action or event facilitated by a vulnerability that results in an unwanted impact to a computer system or application
- Threat actors include:
  - Non-target specific, e.g. computer viruses, worms, trojans and logic bombs.
  - Employees, e.g. disgruntled staff or contractors
  - Organized crime and criminals
  - Corporations, e.g. partners or competitors
  - Human, unintentional (including accidents and carelessness)
  - Human, intentional
  - Natural, e.g. flood, fire, lightning, meteor, earthquakes

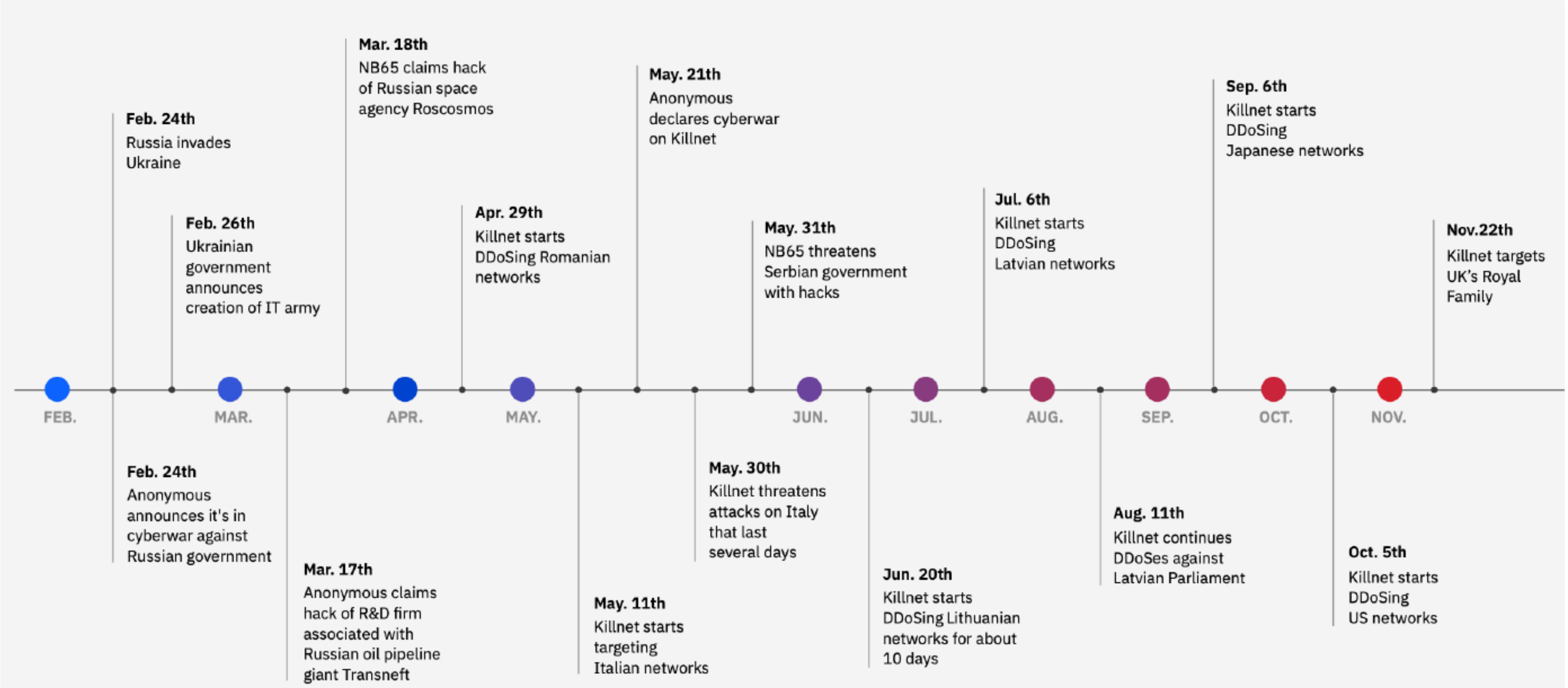
Hackers

- Threat actors that break into computer systems or networks to gain access:
  - White hat hackers break into networks or computer systems to identify any weaknesses so that the security of a system or network can be improved. These break-ins are done with prior permission and any results are reported back to the owner
  - Black hat hackers take advantage of any vulnerability for illegal personal, financial or political gain
  - Gray hat hackers may set out to find vulnerabilities in a system without prior permission of the owner. When they uncover weaknesses, they do not exploit them, rather they report them, but they may demand payment in return
- Unskilled hackers are called script kiddies; they use scripts or programs developed by others, primarily for malicious purposes

Hacktivists

- Hacktivists make political statements to create awareness about issues that are important to them
- In 2022, a significant increase in hacktivist activity has been observed, especially since the start of Russia-Ukraine conflict
- Target organisations through DDoS attacks, defacements and data leaks
- Some of the major Hacktivist groups include Anonymous, TeamOneFirst, GhostSec, Against the West, NB65, KILLNET, XakNet, and The Red Bandits

Timeline of selected Hacktivist Events 2022



Feb. 24th
Russia invades Ukraine

Feb. 24th
Anonymous announces it's in cyberwar against Russian government

Feb. 26th
Ukrainian government announces creation of IT army

Mar. 17th
Anonymous claims hack of R&D firm associated with Russian oil pipeline giant Transneft

Mar. 18th
NB65 claims hack of Russian space agency Roscosmos

Apr. 29th
Killnet starts DDoSing Romanian networks

May. 11th
Killnet starts targeting Italian networks

May. 21th
Anonymous declares cyberwar on Killnet

May. 30th
Killnet threatens attacks on Italy that last several days

May. 31th
NB65 threatens Serbian government with hacks

Jun. 20th
Killnet starts DDoSing Lithuanian networks for about 10 days

Jul. 6th
Killnet starts DDoSing Latvian networks

Aug. 11th
Killnet continues DDoSes against Latvian Parliament

Sep. 6th
Killnet starts DDoSing Japanese networks

Oct. 5th
Killnet starts DDoSing US networks

Nov.22th
Killnet targets UK's Royal Family

Timeline of select hacktivist events 2022
Source: IBM Security X-Force Threat Intelligence Index 2023

Cyber Criminals

- Cyber criminals are usually highly sophisticated and organised
- Main interest in attacks that usually lead to ransomware deployment, coin mining, stealing cryptocurrency, or stealing credentials
- They may even provide cybercrime as a service to other criminals, aka hacker-for hire within the 'Access-as-a-Service' (AaaS) market

State Sponsored Threat Actors

- State-sponsored attackers gather intelligence or commit sabotage on behalf of their government
- They are usually highly trained and well-funded
- Their attacks are focused on specific goals that are beneficial to their government
- Mostly involved in destructive or disruptive operations
- Example: Since the start of the Russia-Ukraine conflict, widespread use of wiper malware attacks to destroy and disrupt networks of governmental agencies and critical infrastructure entities have been observed

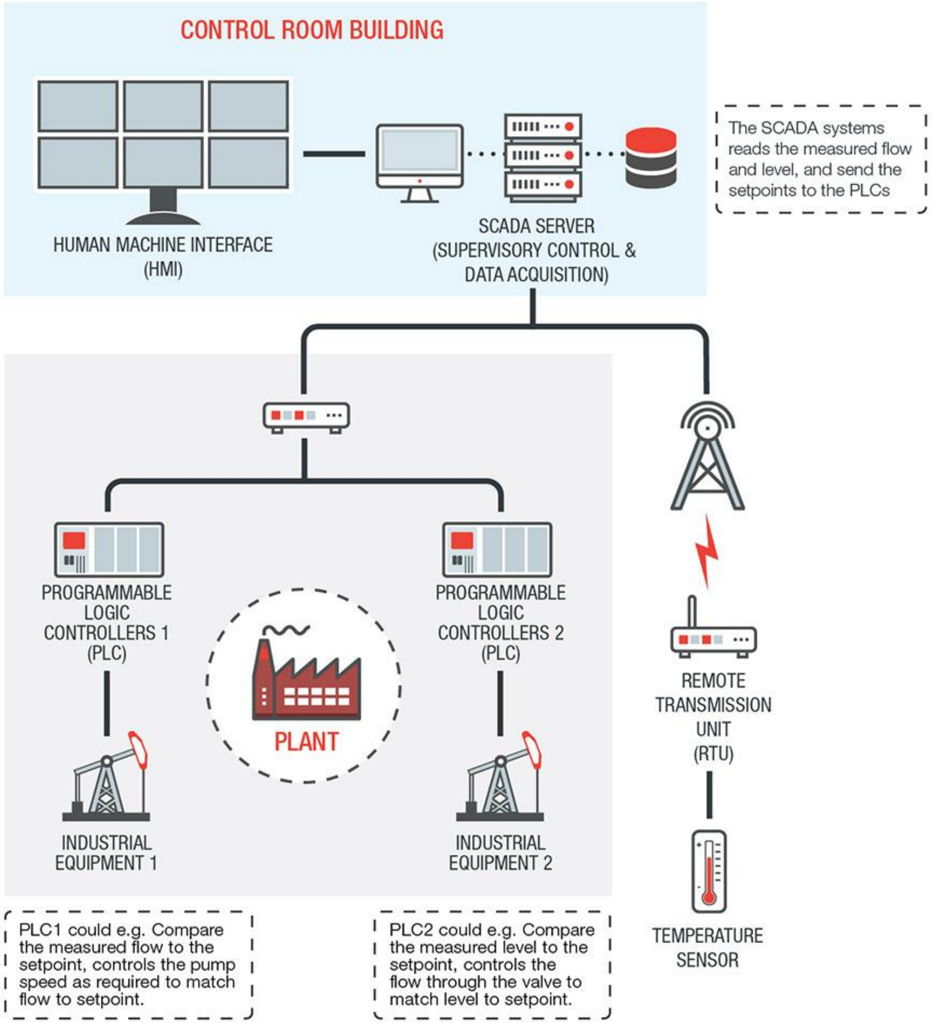
Cyber Warfare

- Cyberwarfare is the use of technology to penetrate and attack another nation's computer systems and networks to cause damage or disrupt critical services
- The main reason for resorting to cyberwarfare is to gain advantage over adversaries
  - Industrial and military espionage e.g. steal defence secrets and gather information about technology
  - Impact infrastructure e.g. power grid
- Example Stuxnet

Some Case Studies

Background: Industrial Control Systems (ICS)

- □ An ICS is an electronic control system and associated instrumentation used for industrial process control
- □ Control systems can range in size from a few modular panel-mounted controllers to large interconnected and interactive distributed control systems (DCSs) with many thousands of field connections
- □ Control systems receive data from remote sensors measuring process variables (PVs), compare the collected data with desired setpoints (SPs), and derive command functions that are used to control a process through the final control elements (FCEs), such as control valves


CONTROL ROOM BUILDING
HUMAN MACHINE INTERFACE (HMI)
SCADA SERVER (SUPERVISORY CONTROL & DATA ACQUISITION)
PROGRAMMABLE LOGIC CONTROLLERS 1 (PLC)
PLANT
PROGRAMMABLE LOGIC CONTROLLERS 2 (PLC)
INDUSTRIAL EQUIPMENT 1
INDUSTRIAL EQUIPMENT 2
REMOTE TRANSMISSION UNIT (RTU)
TEMPERATURE SENSOR

The SCADA systems reads the measured flow and level, and send the setpoints to the PLCs

PLC1 could e.g. Compare the measured flow to the setpoint, controls the pump speed as required to match flow to setpoint.

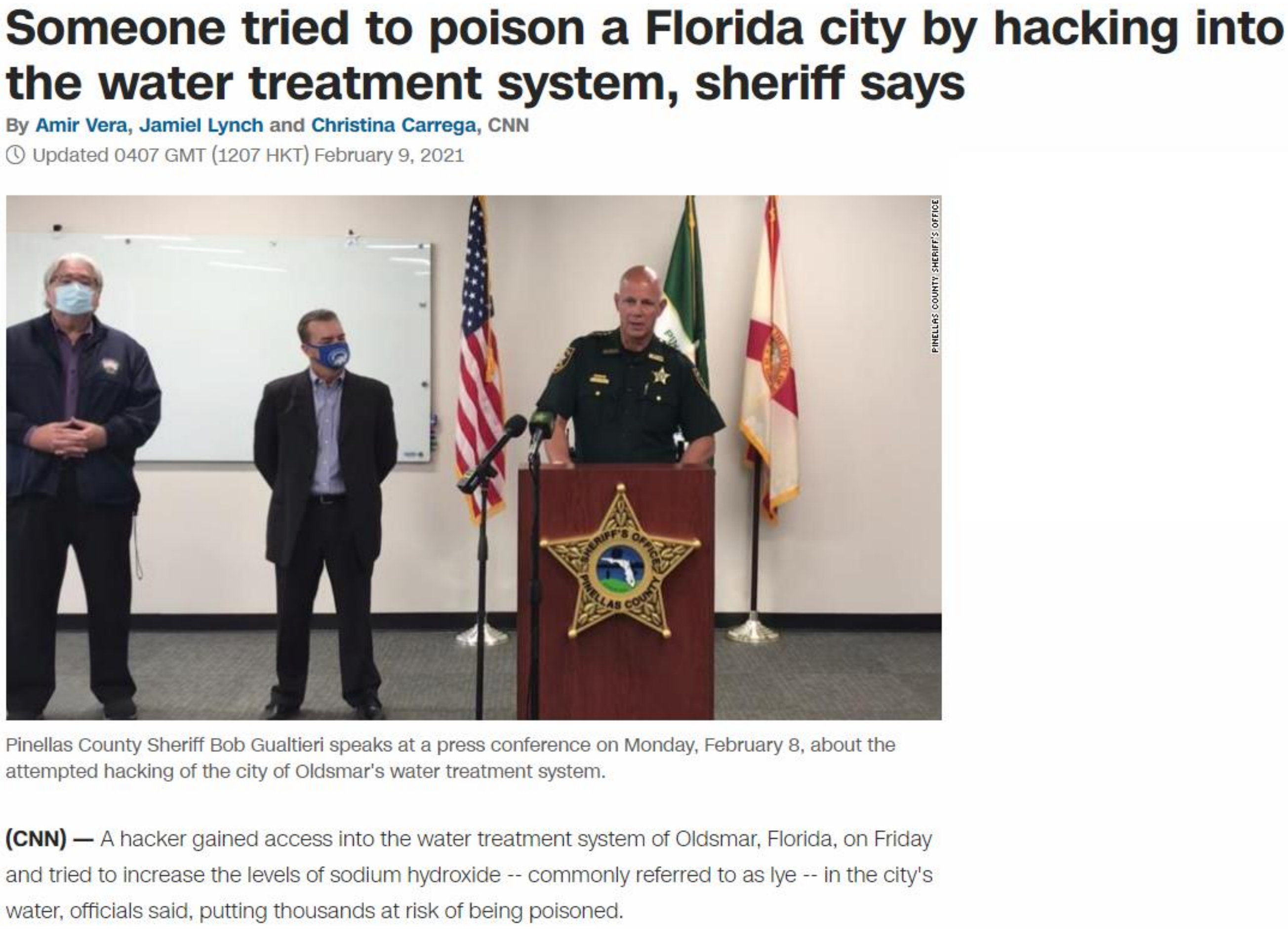
PLC2 could e.g. Compare the measured level to the setpoint, controls the flow through the valve to match level to setpoint.

Cyberattacks on ICS

- Traditionally relatively “niche”, but potentially high-impact attacks on critical infrastructure
  - Power generation
  - Chemical plants
  - Steel mills
  - Transport systems
  - Oil pipelines
- A summary of some recent high-profile attacks can be found here:
  - https://www.makeuseof.com/cyberattacks-on-industry-hackers/

Attacking Critical Infrastructure (Water Supply)

- https://edition.cnn.com/2021/02/08/us/oldsmar-florida-hack-water-poison/index.html?utm_source=twCNN&utm_term=link&utm_medium=social&utm_content=2021-02-09T02%3A55%3A27



Someone tried to poison a Florida city by hacking into the water treatment system, sheriff says

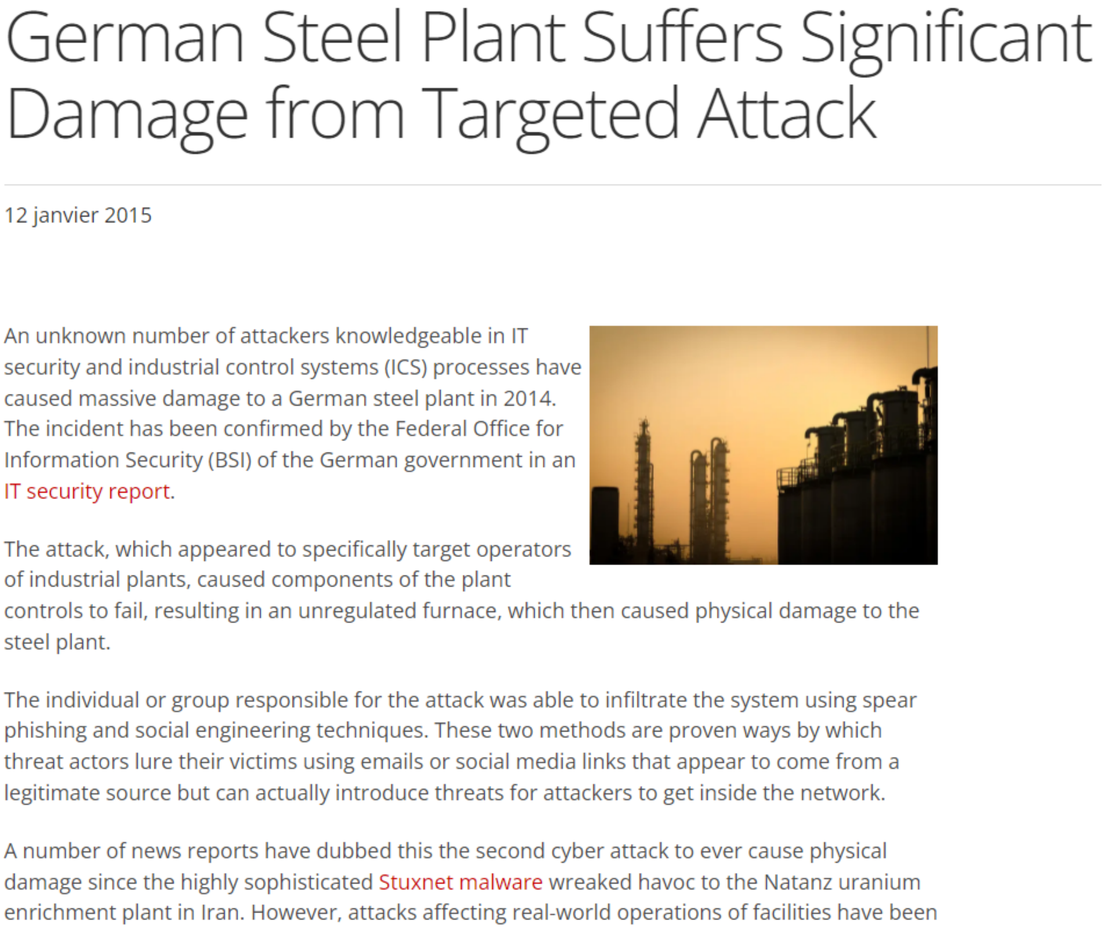
By Amir Vera, Jamiel Lynch and Christina Carrega, CNN

Updated 0407 GMT (1207 HKT) February 9, 2021

Pinellas County Sheriff Bob Gualtieri speaks at a press conference on Monday, February 8, about the attempted hacking of the city of Oldsmar's water treatment system.

(CNN) — A hacker gained access into the water treatment system of Oldsmar, Florida, on Friday and tried to increase the levels of sodium hydroxide -- commonly referred to as lye -- in the city's water, officials said, putting thousands at risk of being poisoned.

Cyberattacks on Industrial Infrastructure



German Steel Plant Suffers Significant Damage from Targeted Attack

12 janvier 2015

An unknown number of attackers knowledgeable in IT security and industrial control systems (ICS) processes have caused massive damage to a German steel plant in 2014. The incident has been confirmed by the Federal Office for Information Security (BSI) of the German government in an IT security report.

The attack, which appeared to specifically target operators of industrial plants, caused components of the plant controls to fail, resulting in an unregulated furnace, which then caused physical damage to the steel plant.

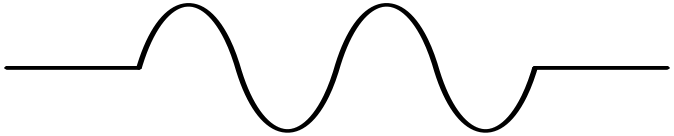
The individual or group responsible for the attack was able to infiltrate the system using spear phishing and social engineering techniques. These two methods are proven ways by which threat actors lure their victims using emails or social media links that appear to come from a legitimate source but can actually introduce threats for attackers to get inside the network.

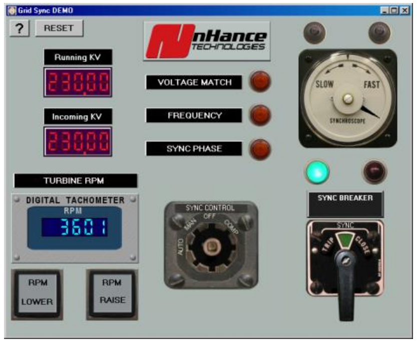
A number of news reports have dubbed this the second cyber attack to ever cause physical damage since the highly sophisticated Stuxnet malware wreaked havoc to the Natanz uranium enrichment plant in Iran. However, attacks affecting real-world operations of facilities have been

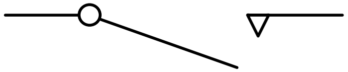
- https://www.trendmicro.com/vinfo/fr/security/news/cyber-attacks/german-steel-plant-suffers-significant-damage-from-targeted-attack

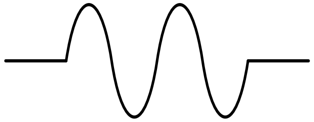
Attacking Critical Infrastructure (Energy Systems) - Synchroscope

- In an electrical grid, power generators and distribution networks must be synced with regard to AC power frequency and phase
- Connecting two unsynchronized AC power systems together is likely to cause high currents to flow, which will severely damage any equipment



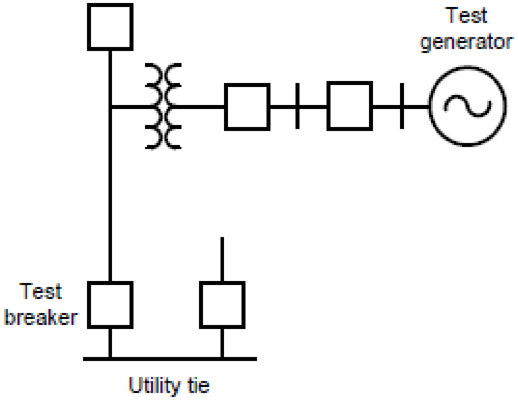






The **AURORA** Cyberattack

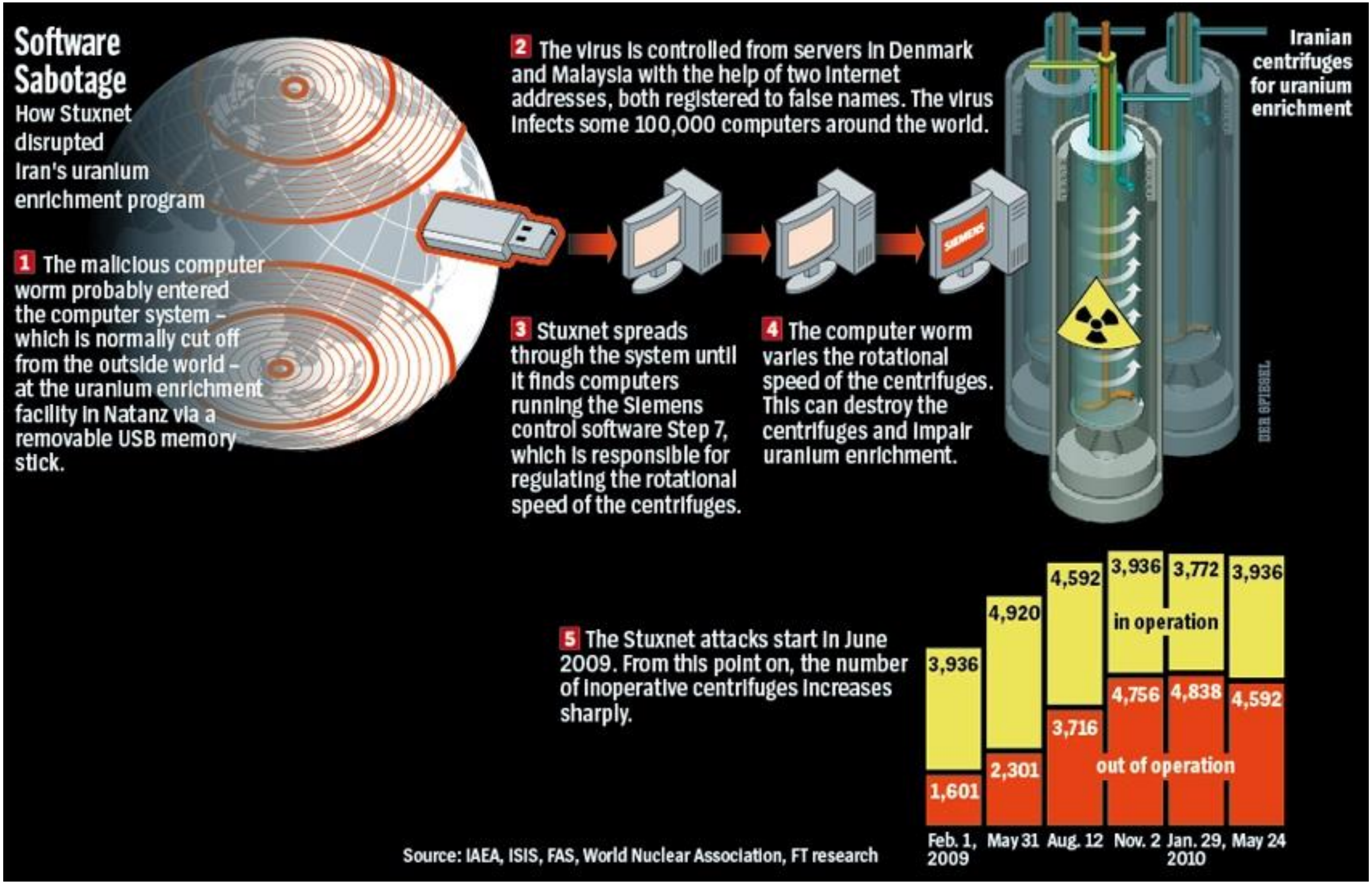
- Experiment conducted by U.S. Department of Energy’s Idaho laboratory
- A hacker gained remote access to a Diesel generator’s control system, see Video,
  - rapidly open and close a diesel generator’s circuit breakers, causing it to become out of sync with the transmission network
  - thereby subjecting the engine to abnormal torques and ultimately causing it to explode
    - high electrical torque translates to stress on the mechanical shaft of the rotating equipment



Example Stuxnet

- Stuxnet was a powerful computer worm designed by U.S. and Israeli intelligence around 2009 designed to disable a key part of the Iranian nuclear program
- It targeted the "air-gapped" nuclear facility at Natanz
- Stuxnet was designed to destroy the centrifuges Iran was using to enrich uranium as part of its nuclear program

Stuxnet



Software Sabotage

How Stuxnet disrupted Iran's uranium enrichment program

1 The malicious computer worm probably entered the computer system – which is normally cut off from the outside world – at the uranium enrichment facility in Natanz via a removable USB memory stick.

2 The virus is controlled from servers in Denmark and Malaysia with the help of two Internet addresses, both registered to false names. The virus infects some 100,000 computers around the world.

3 Stuxnet spreads through the system until it finds computers running the Siemens control software Step 7, which is responsible for regulating the rotational speed of the centrifuges.

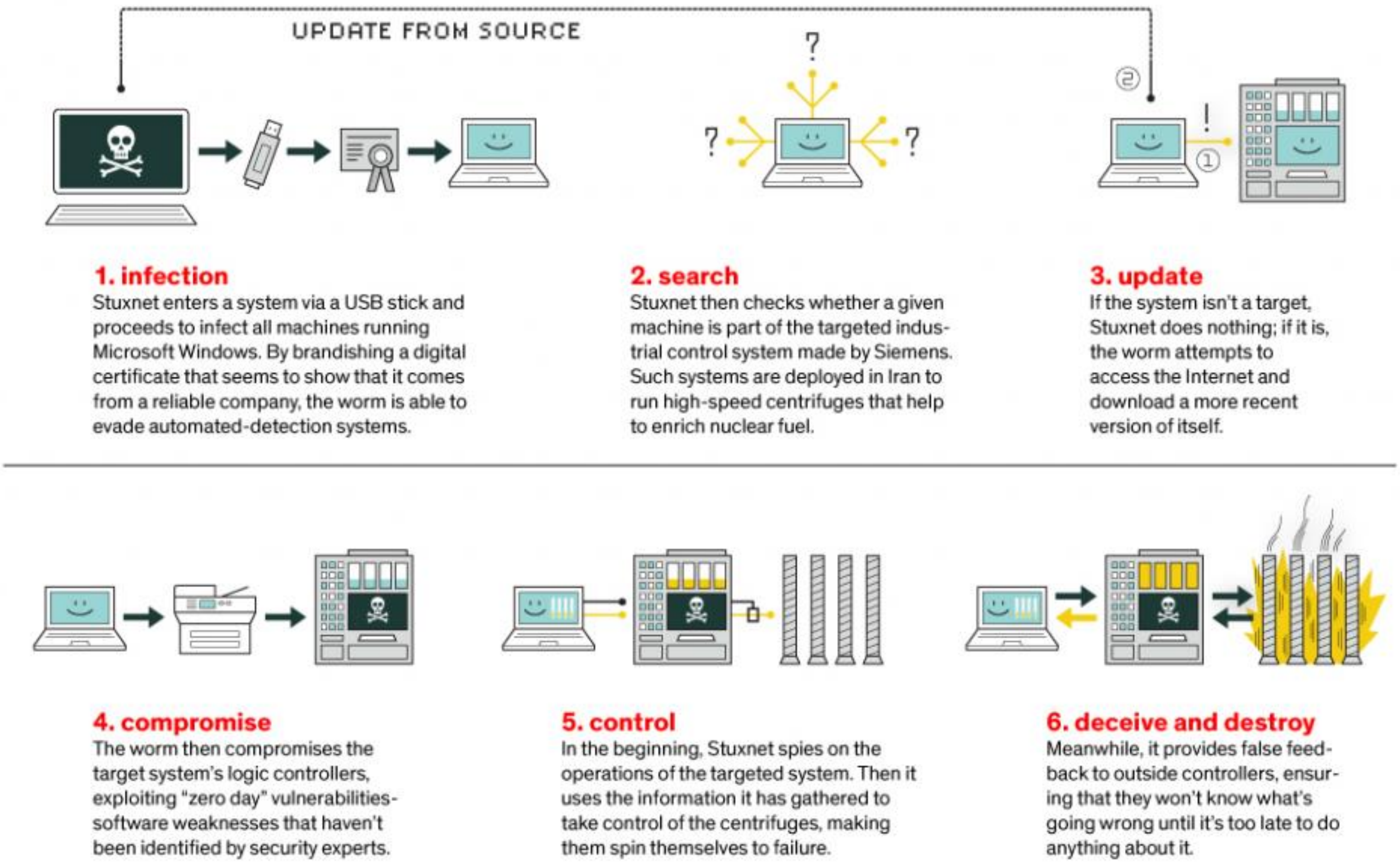
4 The computer worm varies the rotational speed of the centrifuges. This can destroy the centrifuges and impair uranium enrichment.

Iranian centrifuges for uranium enrichment

5 The Stuxnet attacks start in June 2009. From this point on, the number of inoperative centrifuges increases sharply.

Source: IAEA, ISIS, FAS, World Nuclear Association, FT research

Stuxnet Anatomy



1. infection
Stuxnet enters a system via a USB stick and proceeds to infect all machines running Microsoft Windows. By brandishing a digital certificate that seems to show that it comes from a reliable company, the worm is able to evade automated-detection systems.

2. search
Stuxnet then checks whether a given machine is part of the targeted industrial control system made by Siemens. Such systems are deployed in Iran to run high-speed centrifuges that help to enrich nuclear fuel.

3. update
If the system isn't a target, Stuxnet does nothing; if it is, the worm attempts to access the Internet and download a more recent version of itself.

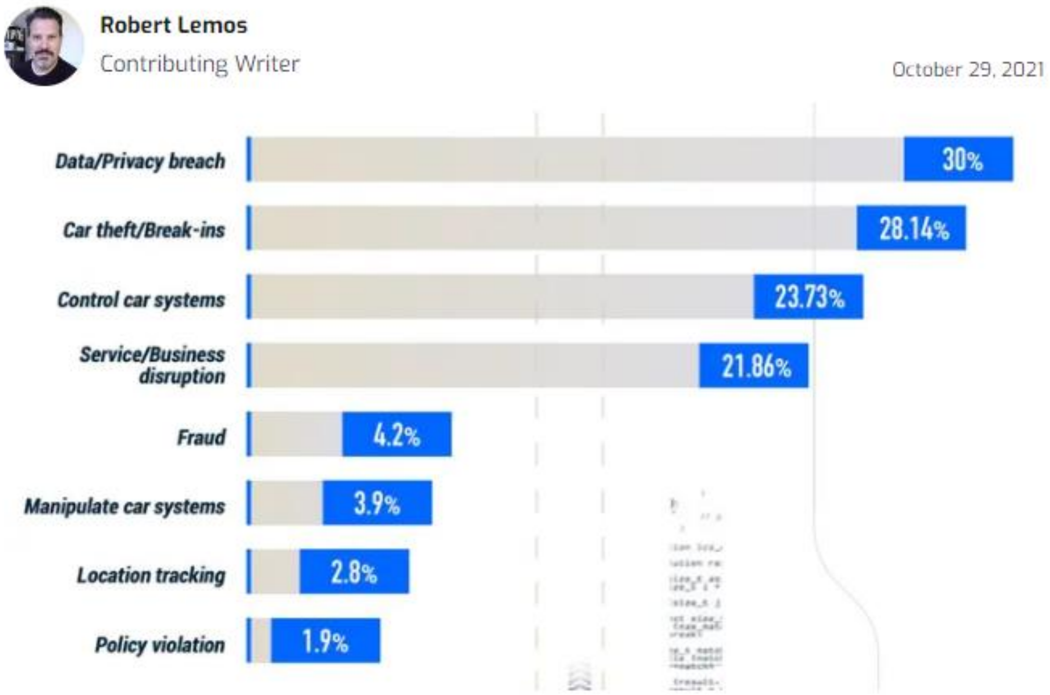
4. compromise
The worm then compromises the target system's logic controllers, exploiting "zero day" vulnerabilities-software weaknesses that haven't been identified by security experts.

5. control
In the beginning, Stuxnet spies on the operations of the targeted system. Then it uses the information it has gathered to take control of the centrifuges, making them spin themselves to failure.

6. deceive and destroy
Meanwhile, it provides false feedback to outside controllers, ensuring that they won't know what's going wrong until it's too late to do anything about it.

Cyberattacks on Connected Cars

- □ https://www.darkreading.com/attacks-breaches/cybercriminals-take-aim-at-connected-car-infrastructure
- □ DEF CON 27: Car Hacking Deconstructed: https://www.youtube.com/watch?v=gzav1K5KSI4



Robert Lemos
Contributing Writer

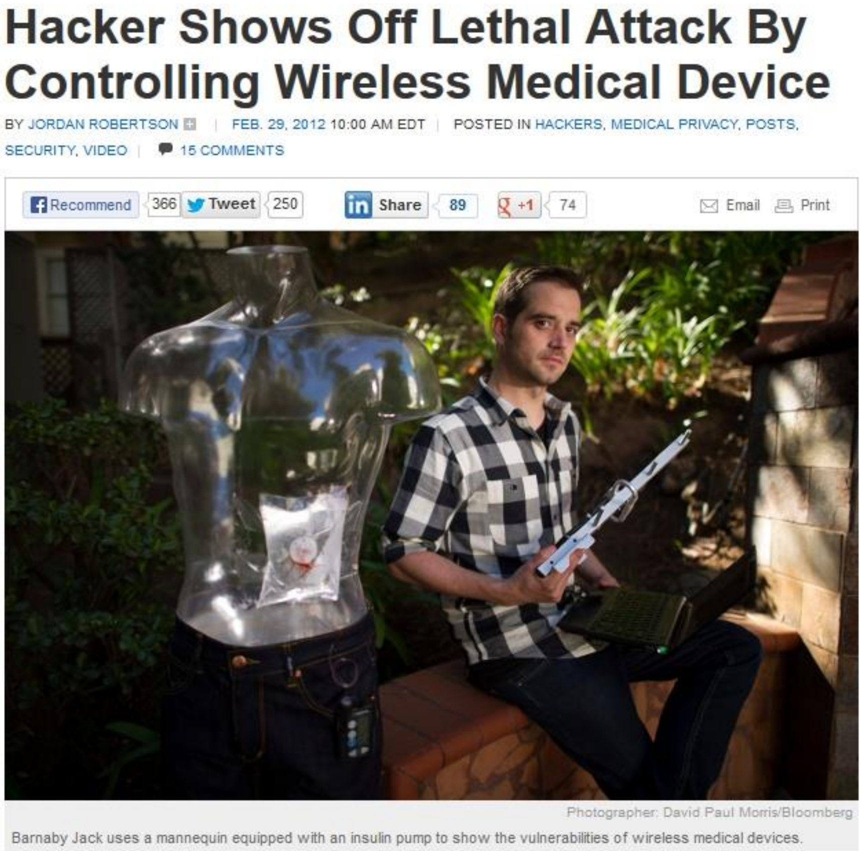
October 29, 2021

Impact of attacks on automakers over the past decade.

Source: Upstream's "Global Automotive Cybersecurity Report 2021"

Cyberattacks on Medical Devices

- Many medical devices are connected to the Internet, or have a wireless interface
- This allows remote attacks, see for example
  - https://www.youtube.com/watch?v=THpcAd2nWJ8



Hacker Shows Off Lethal Attack By Controlling Wireless Medical Device

BY JORDAN ROBERTSON | FEB. 29, 2012 10:00 AM EDT | POSTED IN HACKERS, MEDICAL PRIVACY, POSTS, SECURITY, VIDEO | 15 COMMENTS

Photographer: David Paul Morris/Bloomberg

Barnaby Jack uses a mannequin equipped with an insulin pump to show the vulnerabilities of wireless medical devices.

What is a Vulnerability?

- A weakness which can be exploited by a threat actor/agent (an attacker) to cross privilege boundaries (i.e. perform unauthorised actions) within a computer system (Wikipedia)
- A flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy (RFC2828)
- A weakness in the computational logic (e.g., code) found in software and some hardware components (e.g., firmware) that, when exploited, results in a negative impact to confidentiality, integrity, OR availability (https://cve.mitre.org/)
- Vulnerabilities can be researched, reverse-engineered, hunted, or exploited using automated tools or customized scripts
- An exploitable vulnerability is one for which at least one working attack or exploit exists

Hardware Vulnerabilities

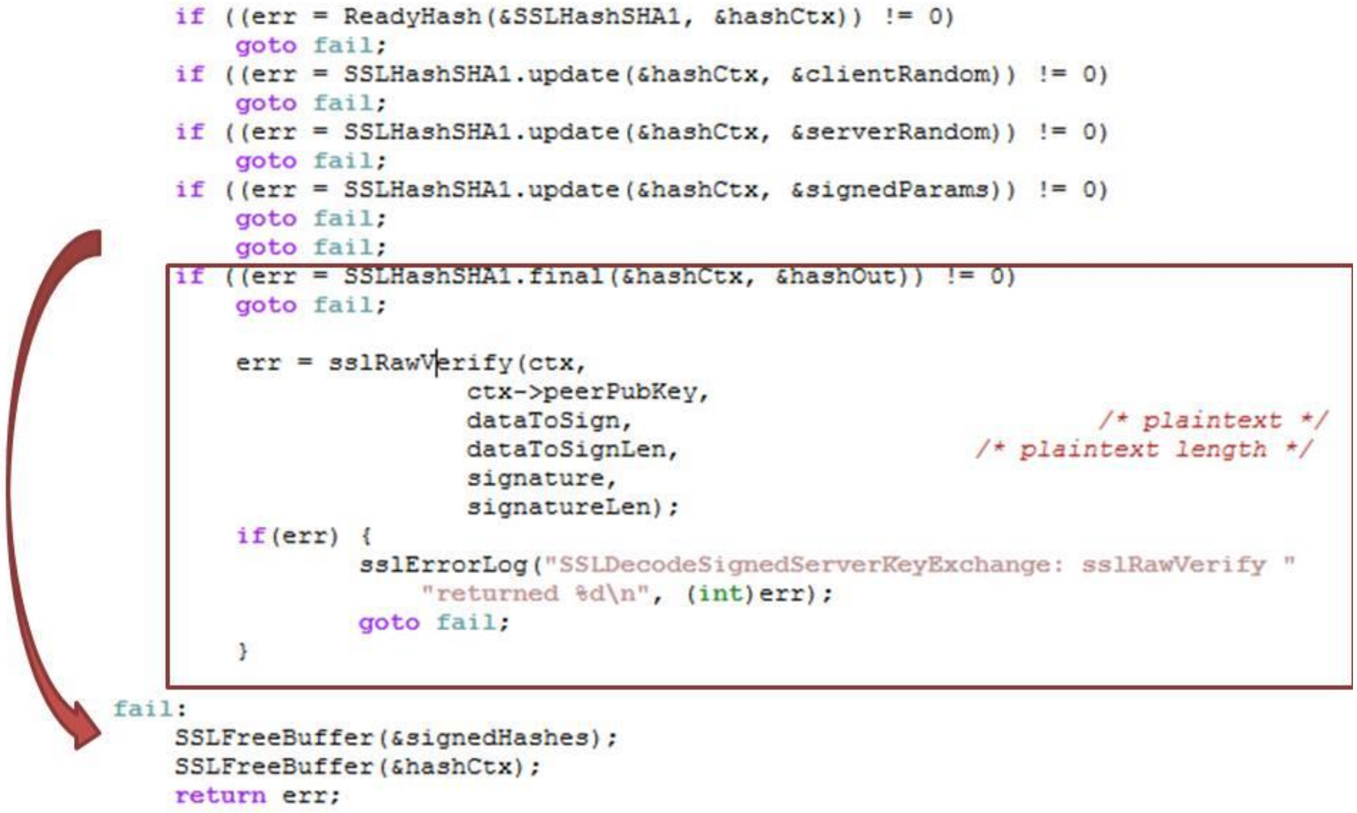
- Hardware vulnerabilities are usually the result of hardware design flaws
- Example DRAM:
  - DRAM memory requires one capacitor per bit (a capacitor is a component which can hold an electrical charge)
  - Modern DRAM chips have a very high memory capacity (4 – 32 gigabits) resulting in those capacitors being positioned installed very close to one another
  - However, it was discovered that due to their close proximity, changes applied to one of these capacitors could influence neighbouring capacitors
  - Based on this design flaw, an exploit called **Rowhammer** was created
  - By repeatedly accessing (hammering) a row of memory, the Rowhammer exploit triggers electrical interferences that eventually corrupt the data stored inside the RAM

Software Vulnerabilities

- Software vulnerabilities are usually introduced by errors in the operating system or application code
- Bug: An error that can be rooted to the source code, e.g.
  - Incorrect implementation of a security protocol
  - Buffer Overflow: When a program writes more data to a buffer than it can hold, potentially leading to arbitrary code execution
    - Example TLS Heartbleed (will be covered later and in an assignment)
- Flaw: An error at a much deeper level, particularly in the design, and likely in the code level, which may be very difficult and costly to correct; e.g.
  - Lack of security features, i.e. data encryption, to protect sensitive application data from unauthorised access

Example: Apple’s ‘goto fail;’ Bug in TLS 1.0 and TLS 1.1 (2014)

- Affected iOS and Mac OS X operation systems
- This vulnerability allowed attacks on TLS connections



```
    if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
        goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;

        err = sslRawVerify(ctx,
                       ctx->peerPubKey,
                       dataToSign,                                /* plaintext */
                       dataToSignLen,                     /* plaintext length */
                       signature,
                       signatureLen);
        if(err) {
                sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify "
                    "returned %d\n", (int)err);
                goto fail;
        }
fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
```

The Common Vulnerabilities and Exposures (CVE) Database

- CVE (https://cve.mitre.org/) is a central repository of all the reported security vulnerabilities associated with a specific software system
- Each CVE entry has a unique identifier which is commonly used by many commercial vulnerability management systems to refer to a specific software vulnerability, e.g.,
  - Heartbleed, CVE-2014-0160
  - “goto fail;”, CVE-2014-1266
- CVE ID Syntax: CVE prefix + Year + Arbitrary Digits

The Common Weakness Enumeration (CWE) Database

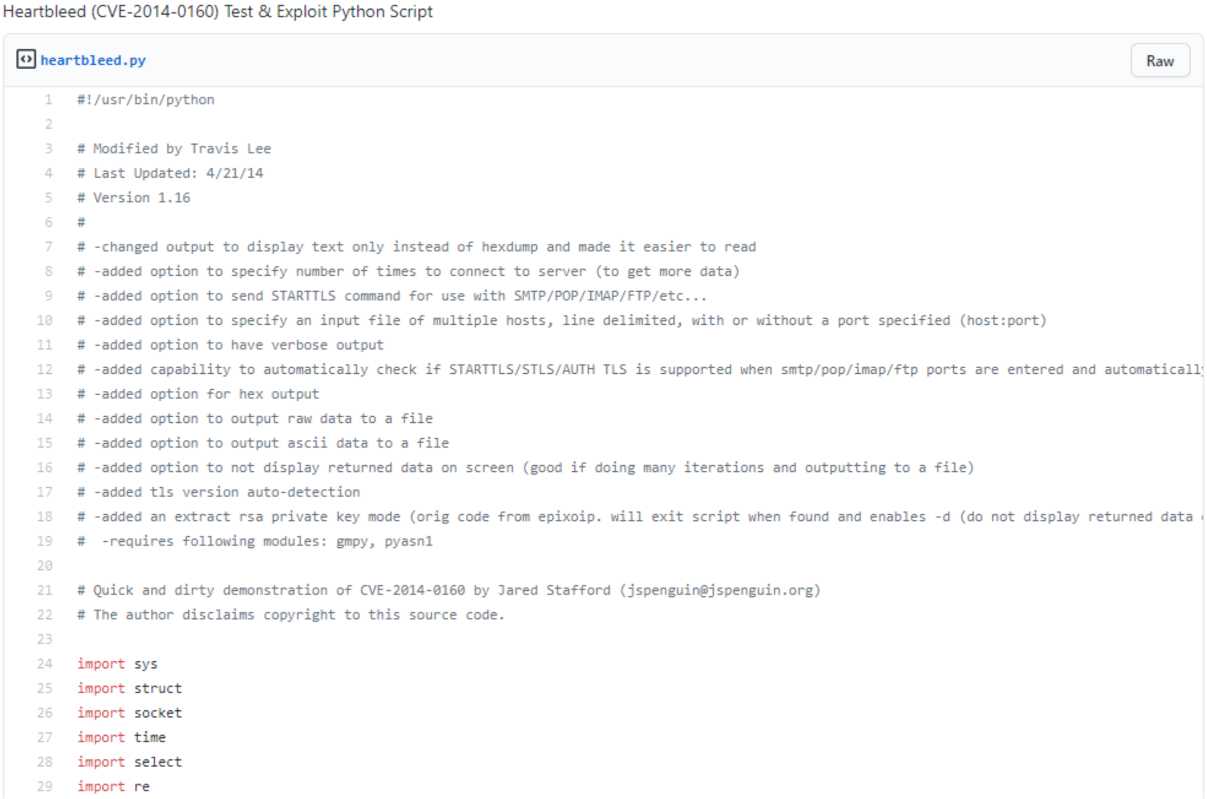
- CVE is complemented by CWE (https://cwe.mitre.org/)
- It provides a formal list of software weakness types that serve as a common language for describing software security weaknesses in architecture, design, or code, for example:
  - Unrestricted upload of files
  - Improper input validation
  - Out-of-bound writes (in arrays)
- CWE describes a generic vulnerability, while CVE has to do with the specific instance within a product or system not the underlying flaw

Exploit (Wikipedia)

- An exploit is a piece of software, data, or a sequence of commands that takes advantage of a vulnerability to cause unintended or unanticipated behaviour to occur on computer software or hardware
- Such behaviour frequently includes things like gaining control of a computer system, allowing privilege escalation, or a denial-of-service attack
- A **remote exploit** works over a network and exploits the security vulnerability without any prior access to the vulnerable system
- A **local exploit** requires prior access to the vulnerable system and usually increases the privileges of the person running the exploit past those granted by the system administrator
- A **zero-day exploit** takes advantage of a vulnerability in software, hardware, or firmware that is unknown to the vendor and for which no patch or fix is available

Heartbleed Exploit Extract (Python Code)

- □ https://gist.github.com/eelsivart/10174134



Heartbleed (CVE-2014-0160) Test & Exploit Python Script

```
#!/usr/bin/python

# Modified by Travis Lee
# Last Updated: 4/21/14
# Version 1.16
#
# -changed output to display text only instead of hexdump and made it easier to read
# -added option to specify number of times to connect to server (to get more data)
# -added option to send STARTTLS command for use with SMTP/POP/IMAP/FTP/etc...
# -added option to specify an input file of multiple hosts, line delimited, with or without a port specified (host:port)
# -added option to have verbose output
# -added capability to automatically check if STARTTLS/STLS/AUTH TLS is supported when smtp/pop/imap/ftp ports are entered and automaticall
# -added option for hex output
# -added option to output raw data to a file
# -added option to output ascii data to a file
# -added option to not display returned data on screen (good if doing many iterations and outputting to a file)
# -added tls version auto-detection
# -added an extract rsa private key mode (orig code from epixoip. will exit script when found and enables -d (do not display returned data
#  -requires following modules: gmpy, pyasn1

# Quick and dirty demonstration of CVE-2014-0160 by Jared Stafford (jspenguin@jspenguin.org)
# The author disclaims copyright to this source code.

import sys
import struct
import socket
import time
import select
import re
```

Attack Surface and Attack Vector

- An organisation’s attack surface is the sum of all its attack vectors, i.e., vulnerabilities, pathways and methods, that hackers can use to gain unauthorised access to a network or sensitive data, or to carry out a cyberattack
- The smaller the attack surface, the easier it is to protect the system (obviously)
- We distinguish between the
  - digital attack surface,
  - physical attack surface (e.g. malicious insiders or device theft),
  - social engineering attack surface (e.g. phishing)

The Digital Attack Surface

- This includes:
  - Weak passwords, i.e., passwords that are easy to guess or easy to crack via brute-force attacks
  - Misconfiguration, e.g., improperly configured network ports or wireless access points
  - Software, operating system and firmware vulnerabilities
  - Outdated or obsolete devices, data, or applications

Social Engineering (Recall CT255)

- Social engineering is the manipulation of people into performing certain actions or revealing confidential information
- That information might be a password, credit card information, personally identifiable information, confidential data, or anything that can be used for fraudulent acts like identity theft
- There are different types of social engineering attacks:
  - Pretexting
  - Phishing
  - Smishing
  - Vishing
  - Tailgating

Pretexting

- □ This is when an attacker calls an individual and lies to them in an attempt to gain access to privileged data
- □ Example:

1. A fraudster **impersonates a trusted authority** and crafts a scenario to reach out to their victims.

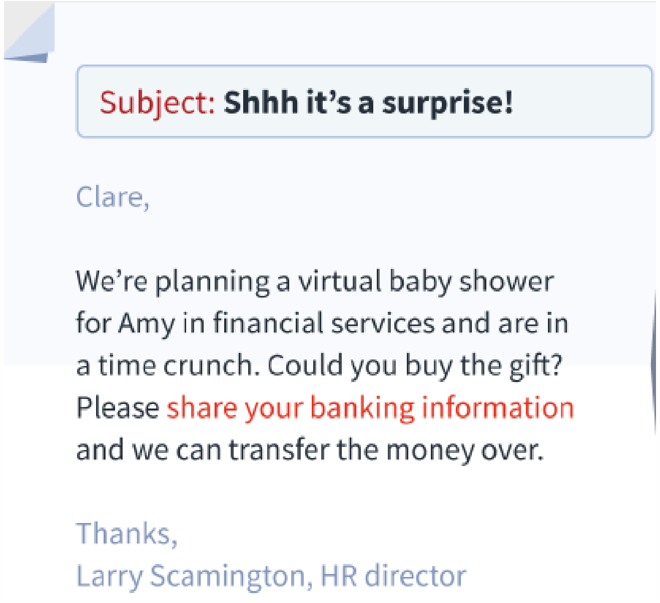
2. The victim **believes the scenario** and shares any information the ‘trusted’ authority requests.

3. The fraudster **gains valuable information** from their victim and often uses it maliciously.

Source: Norton

Phishing

- **Phishing** involves sending malicious emails from supposed trusted sources to as many people as possible, assuming a low response rate (shotgun method)
- In **spear phishing** the perpetrator is disguised as a trusted individual (boss, friend, spouse)
- **Whaling** uses deceptive email messages targeting high-level decision makers within an organisation, such as CEOs and other executives, who have access to highly valuable information



Subject: **Shhh it's a surprise!**

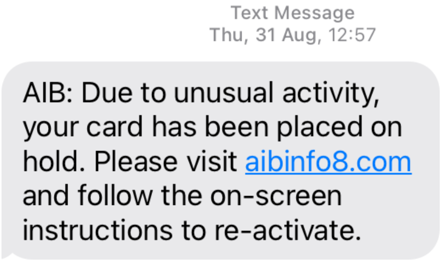
Clare,

We're planning a virtual baby shower for Amy in financial services and are in a time crunch. Could you buy the gift? Please share your banking information and we can transfer the money over.

Thanks,
Larry Scamington, HR director

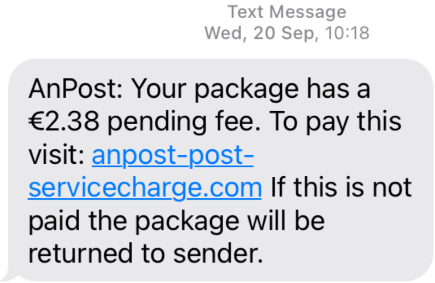
Smishing

- □ Smishing is phishing by SMS or text messaging
- □ This can be a trusty avenue for pretexting attackers to connect with victims since texting is a more intimate form of communication



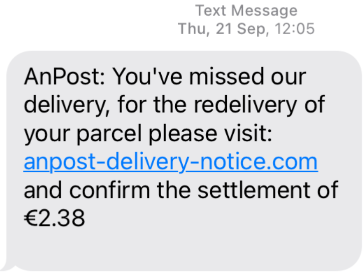
Text Message
Thu, 31 Aug, 12:57

AIB: Due to unusual activity, your card has been placed on hold. Please visit aibinfo8.com and follow the on-screen instructions to re-activate.



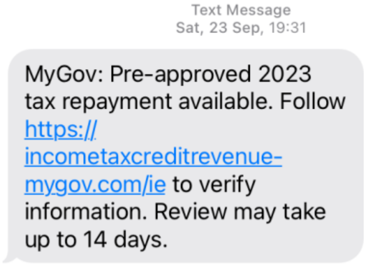
Text Message
Wed, 20 Sep, 10:18

AnPost: Your package has a €2.38 pending fee. To pay this visit: anpost-post-servicecharge.com If this is not paid the package will be returned to sender.



Text Message
Thu, 21 Sep, 12:05

AnPost: You've missed our delivery, for the redelivery of your parcel please visit: anpost-delivery-notice.com and confirm the settlement of €2.38



Text Message
Sat, 23 Sep, 19:31

MyGov: Pre-approved 2023 tax repayment available. Follow https://incometaxcreditrevenue-mygov.com/ie to verify information. Review may take up to 14 days.

Vishing

- It is the voice counterpart to phishing, e.g.
  - An email message asks the user to make a telephone call
  - Victims receive an unsolicited call
- Fraudsters might spoof, or fake caller IDs or use AI generated deepfakes to convince victims they are a trusted source and, ultimately, get victims to share valuable information over the phone
- Many different variations, see for example
  - https://www.youtube.com/watch?v=PWVN3Rq4gzw
  - https://www.youtube.com/watch?v=lc7scxvKQOo

Tailgating

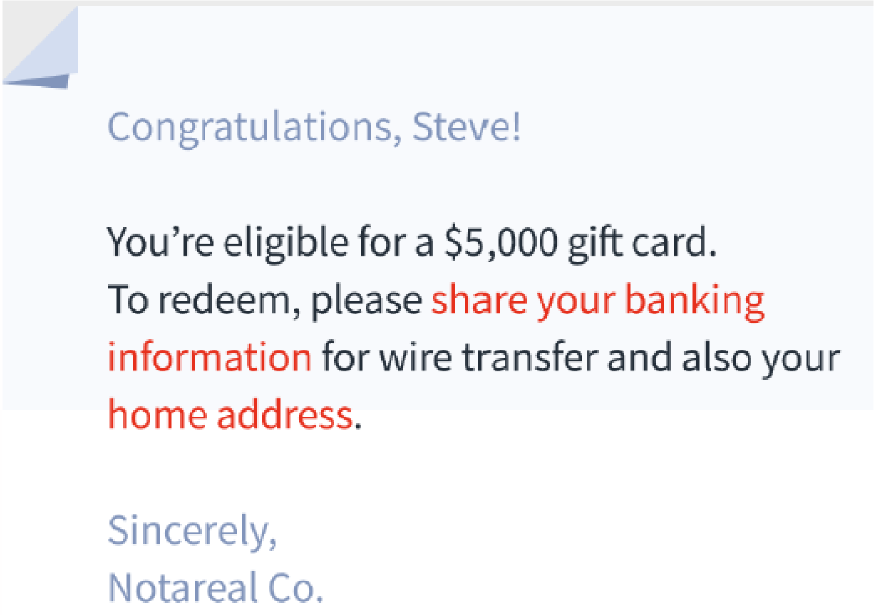
- This is when
  - an attacker quickly follows an authorized person into a secure, physical location
  - fraudsters pose in real-life as someone else to gain access to restricted or confidential areas where they can get their hands on valuable information

An “internet service provider” shows up on your doorstep for a routine check. Once inside, they have free reins to snoop through your devices and valuable information.

TIP
If a service provider arrives without an appointment, don’t just let them inside. Verify their legitimacy by asking questions about your plan.

Quid Pro Quo

- “Get something for doing something” (latin: quid pro quo)
- This is when an attacker requests personal information from a person in exchange for something, like a free gift



Congratulations, Steve!

You’re eligible for a $5,000 gift card.
To redeem, please share your banking information for wire transfer and also your home address.

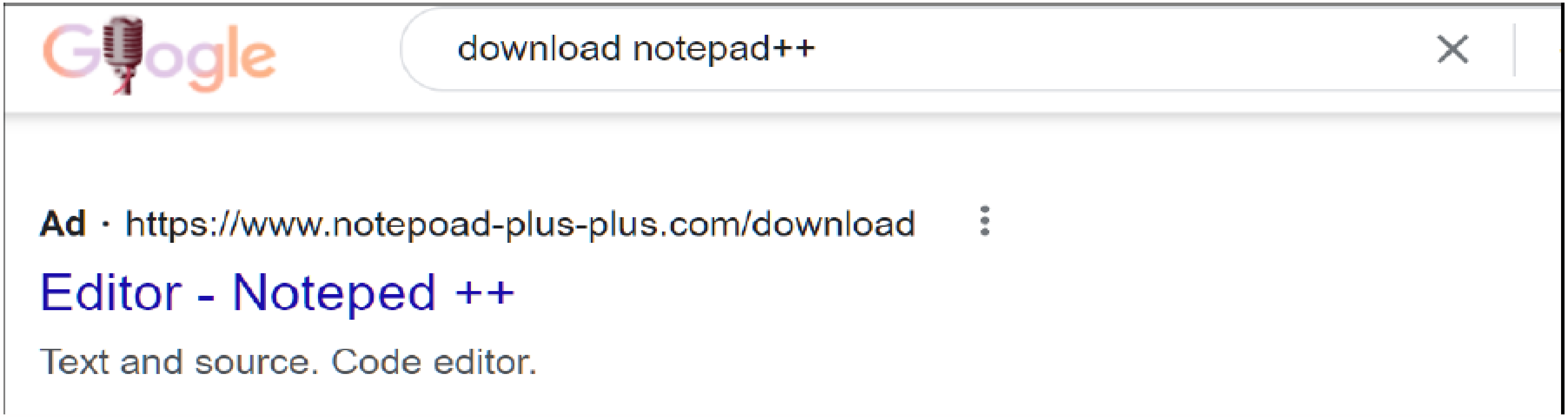
Sincerely,
Notareal Co.

SEO Poisoning

- Search engine optimisation (SEO) is about improving an organisation's website visibility in search engine results
- Search engines such as Google present a list of web pages to users based on their search query. These web pages are ranked according to the relevancy of their content.
- SEO poisoning is a technique used by threat actors to increase the prominence of their malicious websites, making them look more authentic to consumers
- The most common goal of SEO poisoning is to increase traffic to malicious sites that may host malware or attempt social engineering

Typosquatting

- Typosquatting targets users who might open their
- browser and input a website address that has an inadvertent typo or click on a link with a misspelled URL
- § To exploit these minor user errors, attackers register domain names similar to legitimate ones



Ad · https://www.notepoad-plus-plus.com/download

Editor - Noteped ++

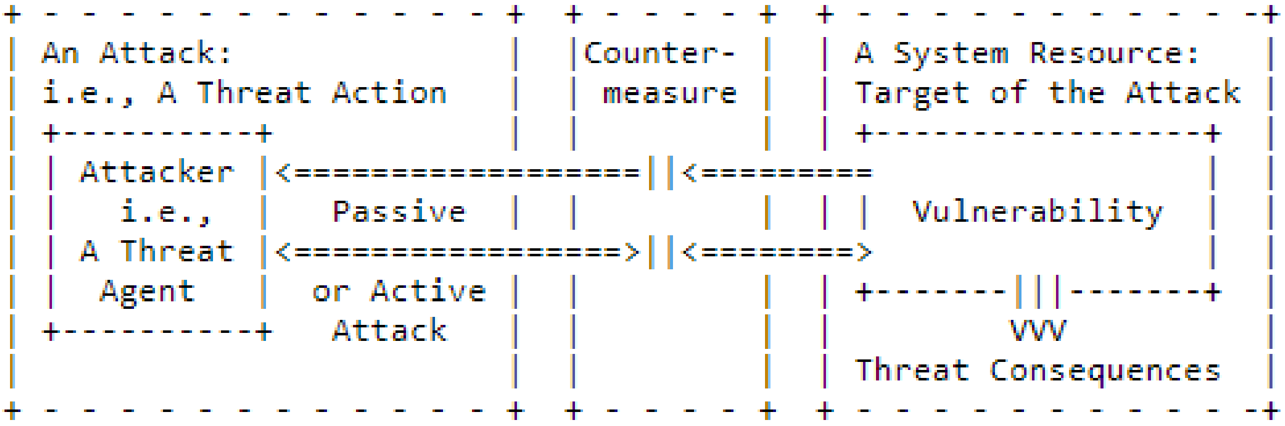
Text and source. Code editor.

Attack (RFC2828, Internet Security Glossary)

□ An attack is an assault on system security that derives from an intelligent threat, i.e. a deliberate attempt

□ An "active attack" attempts to alter system resources or affect their operation (e.g., a ransomware attack on a file server)

□ A "passive attack“ attempts to learn or make use of information from the system, but does not affect system resources (e.g., eavesdropping on an unprotected network connection)



```
+ - - - - - - - - - - - - +  + - - - - +  + - - - - - - - - - - -+
| An Attack:              |  |Counter- |  | A System Resource:   |
| i.e., A Threat Action   |  | measure |  | Target of the Attack |
| +----------+            |  |         |  | +-----------------+  |
| | Attacker |<==================||<=========                 |  |
| |   i.e.,  |   Passive  |  |         |  | |  Vulnerability  |  |
| | A Threat |<=================>||<========>                 |  |
| |  Agent   |  or Active |  |         |  | +-------|||-------+  |
| +----------+   Attack   |  |         |  |         VVV          |
|                         |  |         |  | Threat Consequences  |
+ - - - - - - - - - - - - +  + - - - - +  + - - - - - - - - - - -+
```

Common Attack Types

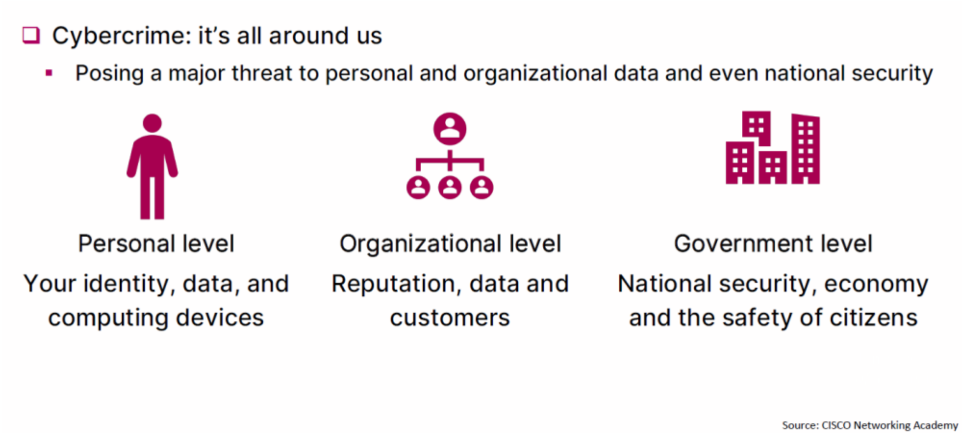
- Malware
  - E.g., in a ransomware attack, an adversary encrypts a victim’s data and offers to provide a decryption key in exchange for a payment
- Denial-of-Service (DoS)
  - A malicious, targeted attack that floods a network with false requests in order to disrupt business operations
- Phishing
  - A type of cyberattack that uses email, SMS, phone, social media, and social engineering techniques to entice a victim to share sensitive information
- Spoofing
  - A technique through which a cybercriminal disguises themselves as a known or trusted source
- Code injection attacks
  - An attacker injecting malicious code into a vulnerable computer or network to change its course of action (e.g. XSS and SQL injection)

Countermeasures (RFC2828, Internet Security Glossary)

- An action, device, procedure, or technique that
  - … reduces a threat, a vulnerability, or an attack
  - … by eliminating or preventing it,
  - … by minimising the harm it can cause, or
  - … by discovering and reporting it so that corrective action can be taken
- For example, a firewall filters unsolicited network traffic

Threat Consequences

- Threat consequences (as a result from an attack) include
  - disclosure of information
  - deception,
  - disruption of services
  - usurpation, e.g. unauthorized control of some part of a system



- Cybercrime: it’s all around us
  - Posing a major threat to personal and organizational data and even national security

Personal level
Your identity, data, and computing devices

Organizational level
Reputation, data and customers

Government level
National security, economy and the safety of citizens

Source: CISCO Networking Academy

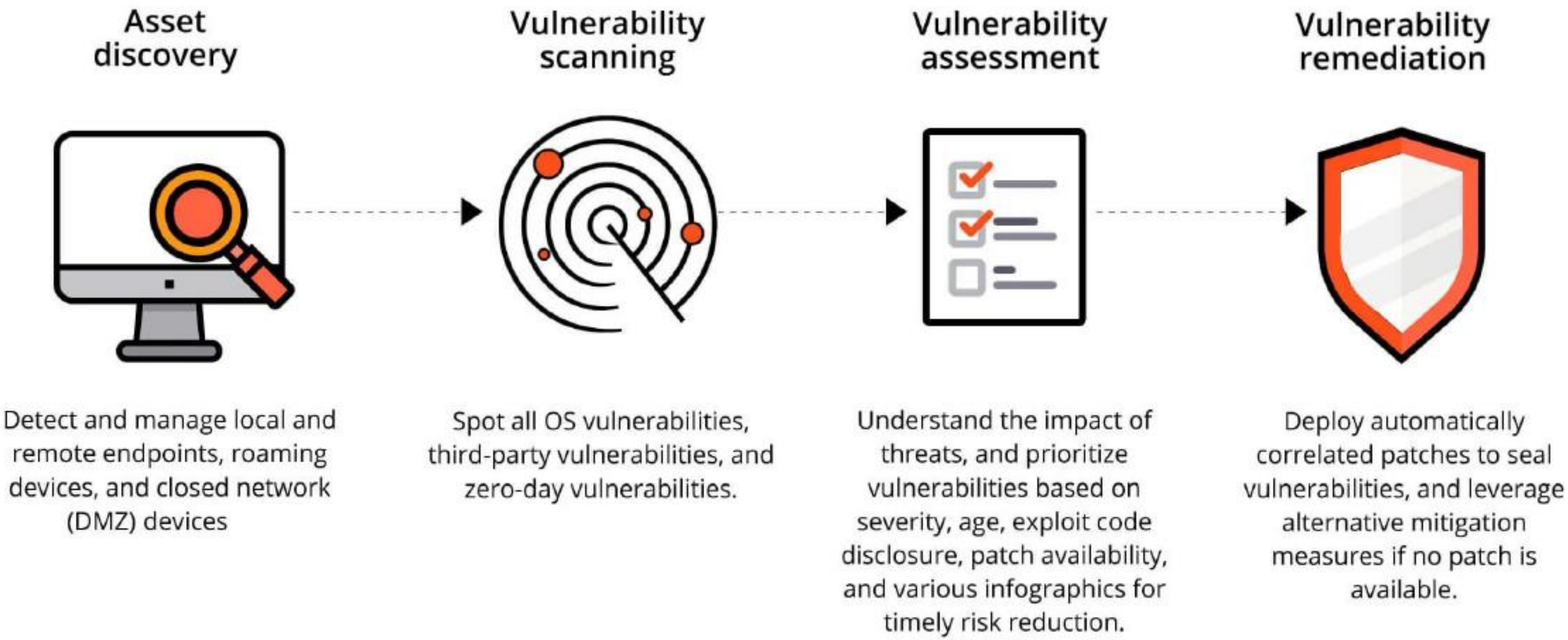
Vulnerability Testing

- Vulnerability testing is a process of evaluating and identifying security weaknesses in a computer system, network, or software application
- It involves systematically scanning, probing, and analyzing systems and applications to uncover potential vulnerabilities, such as coding errors, configuration flaws, or outdated software components
- The main goal of vulnerability testing is to discover and address these security gaps before they can be exploited by attackers

Vulnerability Testing

- **Network-based scanning:** Used to scan networks for open ports, misconfigurations, and other security weaknesses
- **Web application scanning:** Identify vulnerabilities in web applications, such as SQL injection, cross-site scripting (XSS), and broken authentication
- **Static application security testing (SAST):** Analyse source code or compiled code to identify potential security vulnerabilities without executing the application
- **Dynamic application security testing (DAST):** Interact with running applications to identify security weaknesses during runtime
- **Fuzz testing:** Generate and send malformed or unexpected inputs to applications to identify vulnerabilities related to input validation and error handling
- **Database vulnerability assessment:** Scan the database management systems for any potential security weaknesses, misconfigurations, or other vulnerabilities that could be exploited
- **Configuration management and compliance assessment:** Assess system and application configurations against established security best practices or compliance standards
- **Container and cloud security assessment:** Focus on identifying vulnerabilities and misconfigurations in cloud-based environments and containerized applications

Vulnerability Testing Steps



Asset discovery

Detect and manage local and remote endpoints, roaming devices, and closed network (DMZ) devices

Vulnerability scanning

Spot all OS vulnerabilities, third-party vulnerabilities, and zero-day vulnerabilities.

Vulnerability assessment

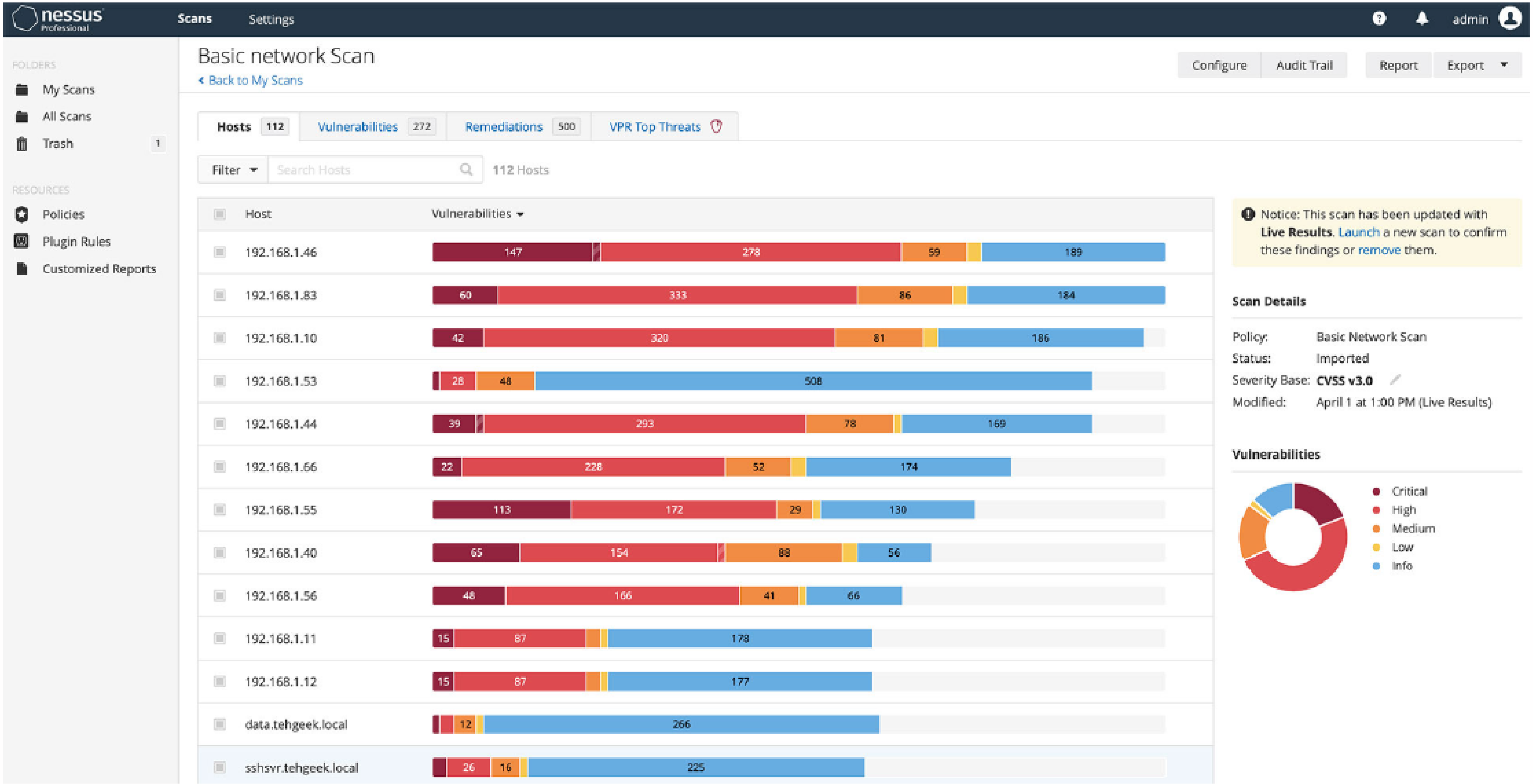
Understand the impact of threats, and prioritize vulnerabilities based on severity, age, exploit code disclosure, patch availability, and various infographics for timely risk reduction.

Vulnerability remediation

Deploy automatically correlated patches to seal vulnerabilities, and leverage alternative mitigation measures if no patch is available.

https://www.manageengine.com/vulnerability-management/vulnerability-assessment.html

Example Nessus: An automatic Network Vulnerability Scanner



Basic network Scan

‹ Back to My Scans

112 Hosts

Host	Vulnerabilities ▾
192.168.1.46	147 278 59 189
192.168.1.83	60 333 86 184
192.168.1.10	42 320 81 186
192.168.1.53	28 48 508
192.168.1.44	39 293 78 169
192.168.1.66	22 228 52 174
192.168.1.55	113 172 29 130
192.168.1.40	65 154 88 56
192.168.1.56	48 166 41 66
192.168.1.11	15 87 178
192.168.1.12	15 87 177
data.tehgeek.local	12 266
sshsvr.tehgeek.local	26 16 225

Notice: This scan has been updated with **Live Results**. Launch a new scan to confirm these findings or remove them.

Scan Details

Policy: Basic Network Scan

Status: Imported

Severity Base: **CVSS v3.0**

Modified: April 1 at 1:00 PM (Live Results)

Vulnerabilities

Outlook Assignment 1

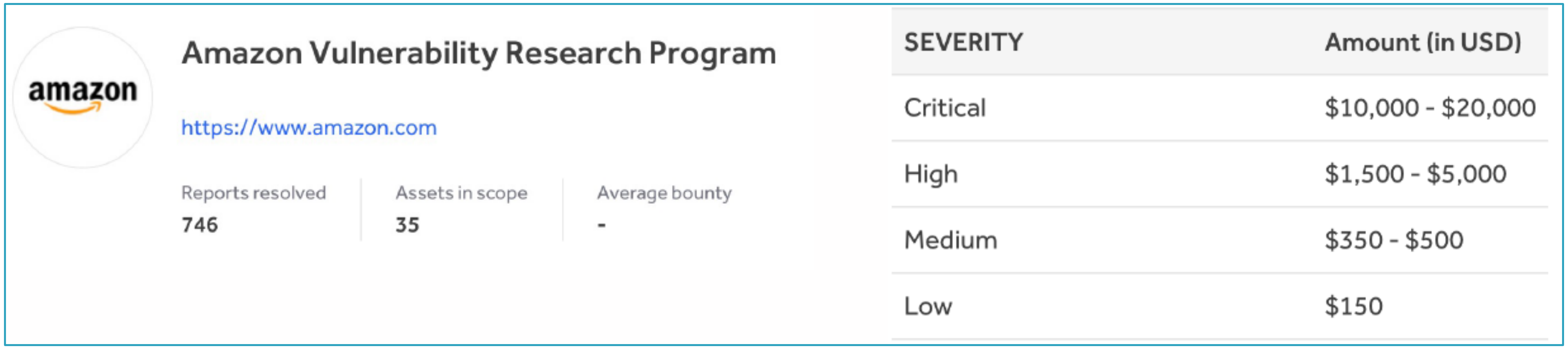
- In assignment 1 you will be doing a manual (non-automated) vulnerability analysis of a VM target, using Metasploit, focusing on a small number of exploits
- Metasploit is a
  - widely-used open-source framework for developing, testing, and executing exploits against target systems
  - powerful tool for penetration testing, enabling security professionals to identify and exploit vulnerabilities in networks, systems, and applications
- This assignment will reinforce your understanding of pentesting tools, vulnerabilities and exploits

Vulnerability Scanning versus Penetration Testing

- Both are essential components of a comprehensive cybersecurity strategy, but they serve different purposes and involve different methodologies
  - The primary goal of **vulnerability scanning** is to identify known vulnerabilities in systems, applications, and networks; it provides an automated way to check for security weaknesses
  - The primary goal of **penetration testing** is to simulate real-world attacks to assess the security posture of a system, application, or network, thereby determining the impact and the effectiveness of existing security measures
    - Pentesters use a combination of automated tools and manual techniques (e.g. social engineering) to find and exploit vulnerabilities by mimicing the actions of real attackers

Vulnerability Disclosure Options

- Tell no one (No disclosure)
- Report in full to public immediately (Full disclosure)
- Report to vendor only and potentially receive bug bounty!



Amazon Vulnerability Research Program

https://www.amazon.com

Reports resolved	Assets in scope	Average bounty
746	35	-

SEVERITY	Amount (in USD)
Critical	$10,000 - $20,000
High	$1,500 - $5,000
Medium	$350 - $500
Low	$150

- Report to vendor, wait for fix, report to public (Responsible disclosure)
- Sell vulnerability to middleman and don't report to vendor
- Develop fully weaponized malware and distribute on black market

CT437 COMPUTER SECURITY AND FORENSIC COMPUTING

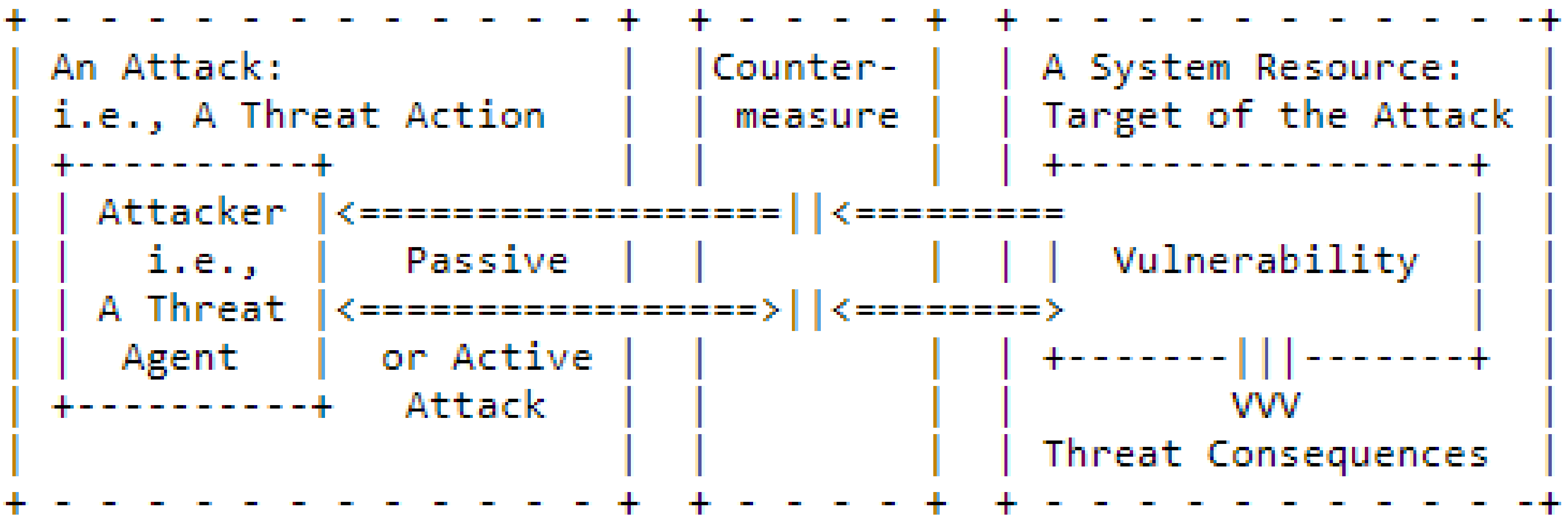
THE CIA TRIAD
REVISION: GDPR AND RAID

Dr. Michael Schukat

Ollscoil na Gaillimhe
University of Galway

Recap: RFC2828

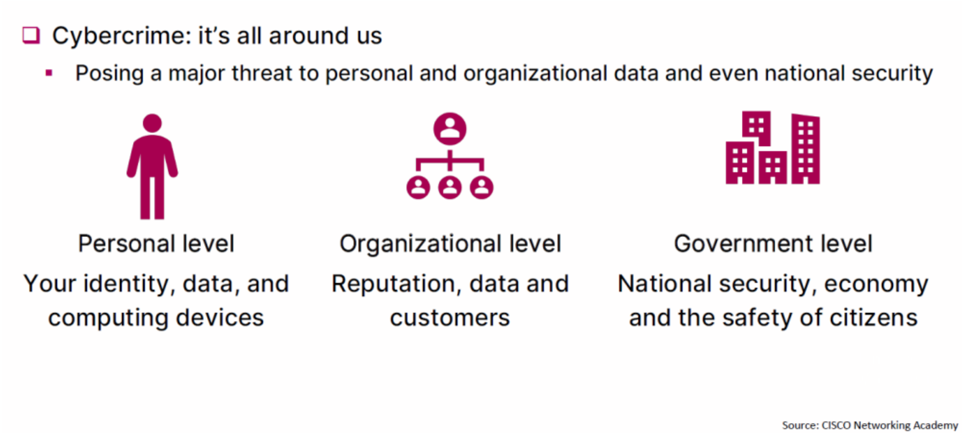
- RFC2828, Internet Security Glossary
- https://tools.ietf.org/html/rfc2828



```
+ - - - - - - - - - - - - +  + - - - - +  + - - - - - - - - - - -+
| An Attack:              |  |Counter- |  | A System Resource:   |
| i.e., A Threat Action   |  | measure |  | Target of the Attack |
| +----------+            |  |         |  | +-----------------+  |
| | Attacker |<==================||<=========                 |  |
| |   i.e.,  |   Passive  |  |         |  | |  Vulnerability  |  |
| | A Threat |<=================>||<========>                 |  |
| |  Agent   |  or Active |  |         |  | +-------|||-------+  |
| +----------+   Attack   |  |         |  |         VVV          |
|                         |  |         |  | Threat Consequences  |
+ - - - - - - - - - - - - +  + - - - - +  + - - - - - - - - - - -+
```

Recap: Threat Consequences

- Threat consequences (as a result from a threat action) include
  - disclosure of information
  - Deception (i.e. pretending to be another entity)
  - disruption of services
  - usurpation, e.g. unauthorized control of some part of a system



- Cybercrime: it's all around us
  - Posing a major threat to personal and organizational data and even national security

Personal level
Your identity, data, and computing devices

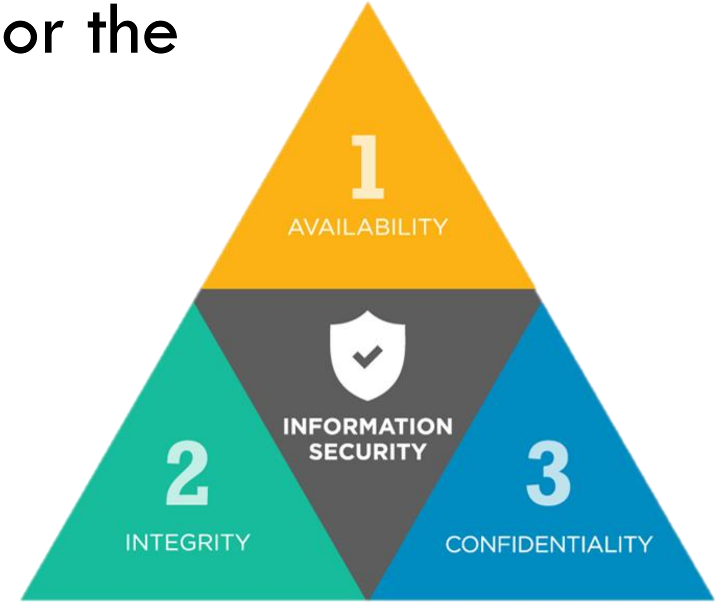
Organizational level
Reputation, data and customers

Government level
National security, economy and the safety of citizens

Source: CISCO Networking Academy

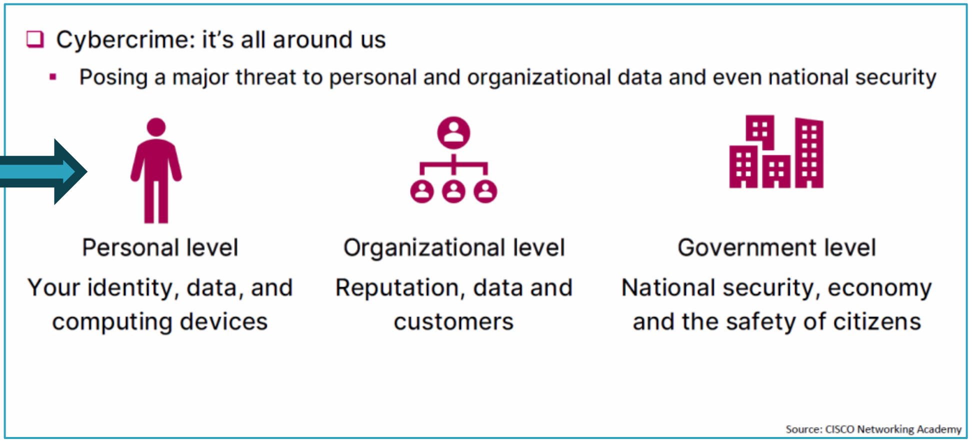
The CIA Triad

- The three letters in "CIA triad" stand for
  - Confidentiality
  - Integrity, and
  - Availability
- It is a model that forms the basis for the development of security systems



CIA Triad: Confidentiality

- □ *“Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information”*
- □ In layman's terms: Keeping things secret that are meant to be secret
- □ Protecting data at rest, in transit, and during processing / use



❑ Cybercrime: it’s all around us

- Posing a major threat to personal and organizational data and even national security

Personal level	Organizational level	Government level
Your identity, data, and computing devices	Reputation, data and customers	National security, economy and the safety of citizens

Source: CISCO Networking Academy

Recall GDPR: Personal Data

- Any information relating to an identified or identifiable natural person ('data subject’)
- An identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person"

Recall GDPR: Sensitive Personal Data

- This includes
  - Racial origin
  - Political opinions
  - Religious or philosophical beliefs
  - Trade Union membership
  - Genetic data (e.g. biological samples)
  - Biometric data (e.g. fingerprints)
  - Data concerning health
  - Data concerning a person's sex life or sexual orientation

- GDPR requires explicit consent to process special categories of personal data

Compromising Confidentiality

- Confidentiality can be compromised by various attacks, including:
  - Man-in-the-middle (MitM) attack
  - Escalation of privileges
  - Human error (weak password, sharing credentials etc.)
  - Insufficient security controls

- Can you identify situations where any of those attacks would apply?

Technologies used to ensure Confidentiality

- These include:
  - Encryption (obviously)
  - Access Control (e.g. multi-factor authentication)
  - Secure network protocols

- Can you name a technology / protocol / algorithm that ensures confidentiality?

PASSWORD
******
MULTIFACTOR AUTHENTICATION
LOGGED IN
NIST National Institute of Standards and Technology U.S. Department of Commerce
www.nist.gov

CIA Triad: Integrity

- “Guarding against improper information modification or destruction, and includes ensuring information non-repudiation and authenticity”
  - Non-repudiation ensures that a party cannot deny having sent or received a message or transaction
  - Authenticity ensures that information and communication come from a trusted source; this includes protecting against impersonation, spoofing and other types of identity fraud
- In layman’s terms: keeping information accurate, complete, and protected from unauthorised modification
- Integrity makes sure that data is trustworthy and not tampered with
- Think of Revolut; can you provide an example for an attack on data integrity?

Technologies to protect Integrity

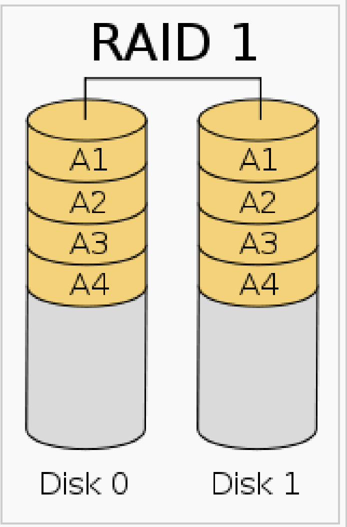
- These include:
  - Network protocols that validate all data exchanged between end points
  - Digital signatures
  - Data hashes
  - Backup and data recovery strategies
  - Version control, to prevent the accidental change or deletion of information

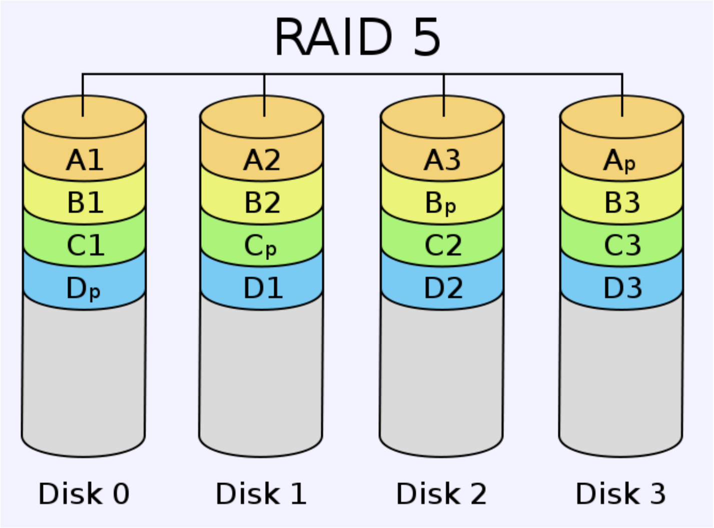
CIA Triad: Availability

- "Ensuring timely and reliable access to and use of information"
- If data is kept confidential and its integrity maintained but it is not available to use, then it is often useless
- Availability can be compromised by various attacks, including:
  - (Distributed) Denial-of-service (DoS) attacks
  - Ransomware
  - Server overload
  - Physical incident such as a power outage or natural disaster

Technologies to provide Availability

- These include:
  - RAID – Redundant Array of Independent Disks
  - Load balancers
  - Business continuity and disaster recovery plans, e.g., redundancy, failover, etc.



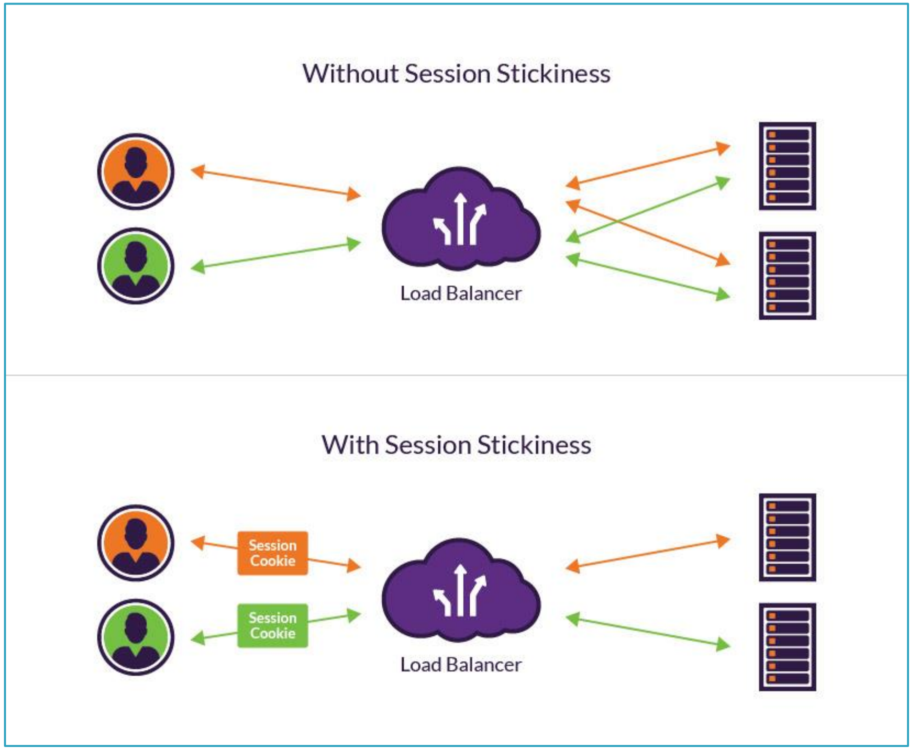


Load Balancers

- Load balancers are server-side gateways that distribute client traffic between multiple backend (e.g., web-) servers
- They require load-balancing cookies on the client side that associate a client session with a particular server, aka **session stickiness**
- A load balancer **creates an affinity** between a client and a specific network server for the duration of a session using a cookie with a random and unique tracking id
- Subsequently, for the duration of the session, the load balancer routes all of the requests of this client to a specific backend server using the tracking id
- GDPR allows the unsolicited use of such cookies via the **communications exemption**

Load Balancers

- Top image:
  - No load balancing at all
- Bottom image:
  - The LB generates and returns a tracking cookie back to a client when its first session is initiated
  - This cookie is tagged to every subsequent client request and allows the LB to forward the request to always the same server (therefore the stickiness)



Data Breaches

- Despite the best of intentions and all the safeguards one can put in place, protecting organisations from every possible cyber attack may not be feasible
- There is an on-going ”arms-race” with cybercriminals constantly finding new ways to attack systems and, very often, they will succeed
- “*A data breach is defined as any breach of security leading to the accidental or unlawful destruction, loss, alteration or unauthorised disclosure of or access to personal data transmitted, stored or otherwise processed*” (article 4.12 GDPR).

CIA Triad Dependencies

1
AVAILABILITY
INFORMATION SECURITY
2
INTEGRITY
3
CONFIDENTIALITY

- Each element connects with the others, and when you implement measures to ensure the protection of one, you must consider the ramifications it has elsewhere
- Example:
  - As a result of the recent cyberattack UoG implemented multi-factor authentication to access all services (email, student records, etc.)
  - Doing so protects the confidentiality of sensitive data, making it harder for unauthorised actors to compromise an employee's login credentials and view information using their account
  - However, without their mobile phone at hand, an employee can't complete the authentication process
  - This hampers therefore the availability of UoG services

Risk Assessment

- ISO 27001 certification, GDPR compliance and other frameworks require the adoption of the CIA triad within an organisation
- All these standards mandate that organisations analyse their operations to measures the risks, threats and vulnerabilities in their systems that could compromise sensitive information
- This process is called **risk assessment**
- We may cover risk assessment at a later stage…

Appendix / Revision

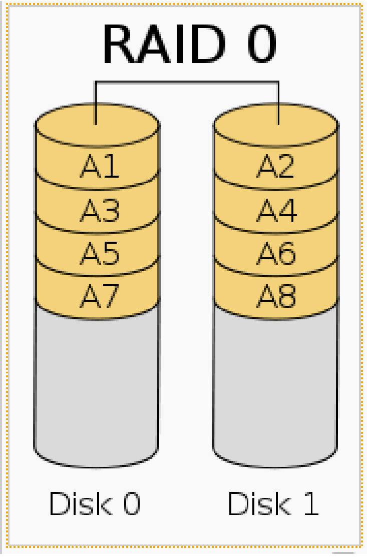
- RAID
- GDPR

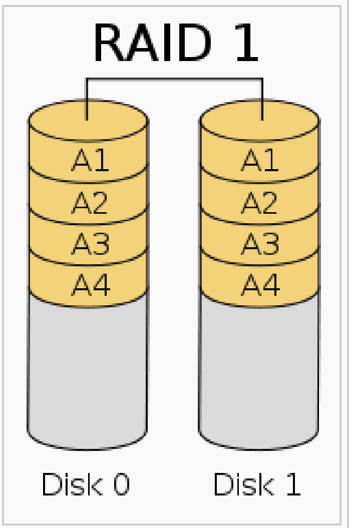
Mass-Storage Redundancy via RAID

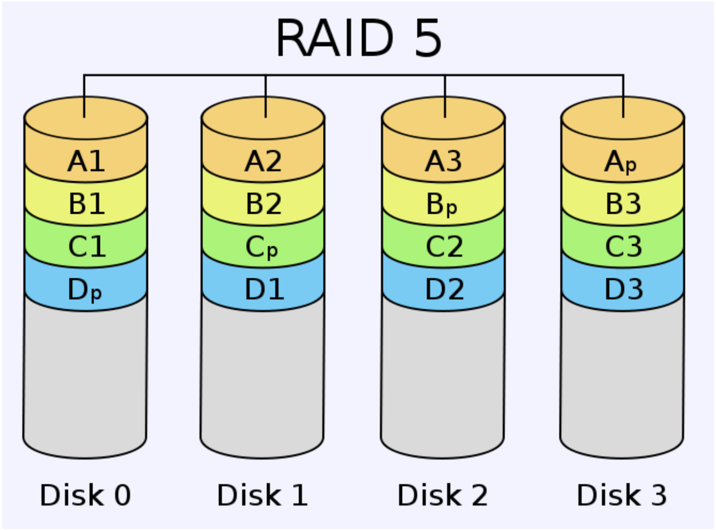
- Background: Hard disks are relatively slow (i.e. seek time + rotational latency) and as mechanical devices may fail
- Redundant Array of Independent Disks (RAID) is a data storage virtualisation technology that combines multiple physical disk drive components into one or more logical units for the purposes of data redundancy and / or performance improvement
- Data blocks are distributed across the drives in one of several ways, referred to as **RAID levels**, depending on the required level of redundancy and performance
- Many RAID levels use a parity-based error protection scheme (see RAID-4), example (with 12 bit / block):
  - Block 1: 010001101001
  - Block 2: 110011011010
  - Block 3: 000100100101
  - P Block: 100111010110 (bitwise EXOR, equivalent to even parity)

Mass-Storage Redundancy via RAID

- RAID storage systems require a dedicated RAID controller, that supports the required RAID level
  - See also the diagram on the next slide
  - Normally such controllers are not shown in RAID diagrams

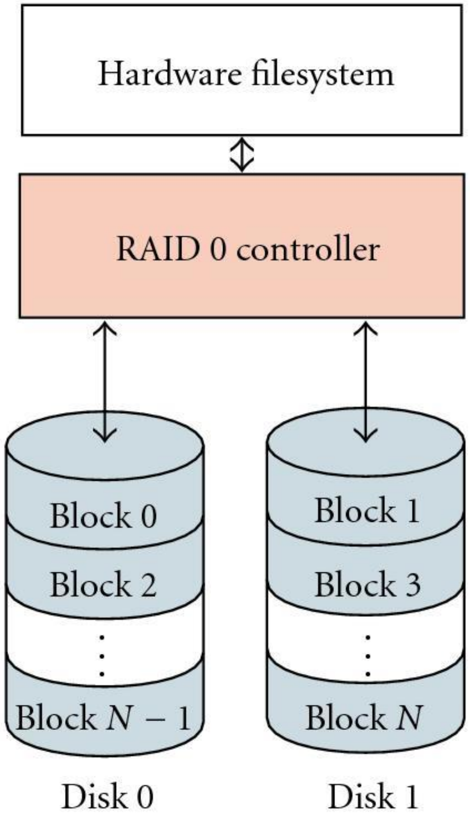






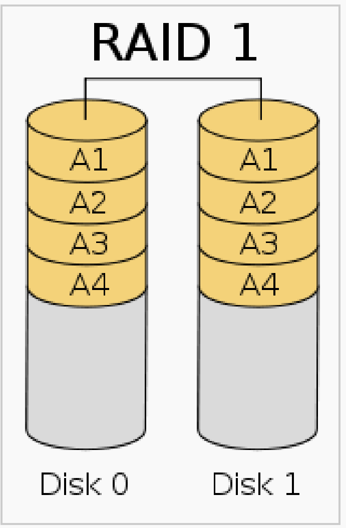
RAID 0

- Block-level striping without parity or mirroring
  - **data striping** is the technique of segmenting logically sequential data, such as a file, so that consecutive segments are stored on different physical storage devices
- 2 or more drives (n) required
- **No redundancy**, but up to n-times R/W performance increase

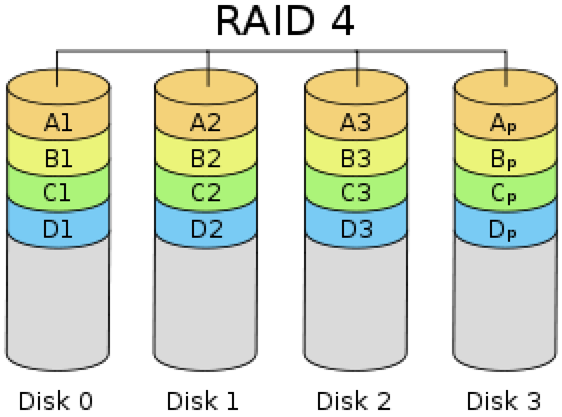


RAID 1

- Block-level mirroring without parity or striping
- 2 or more drives (n) required
- (n – 1) drive failures can be compensated; here each disk can
  - diagnose catastrophic failures (e.g. head crash)
  - detect (but **not** correct) sector-wise bit errors on platters
- No increase in R/W performance



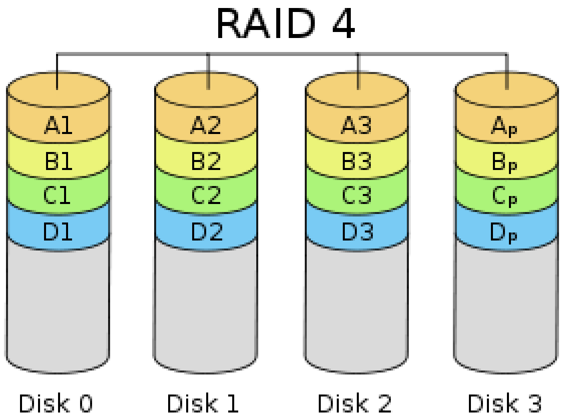
RAID 4



- □ Block-level striping with single parity disk
- □ Single catastrophic drive failure can be compensated (any drive can fail)
- □ RAID 4 provides good performance of random reads, while the performance of random writes is low due to the need to write all parity data to a single disk (Disk 3 in the diagram above)
- □ Minimum of 3 drives required

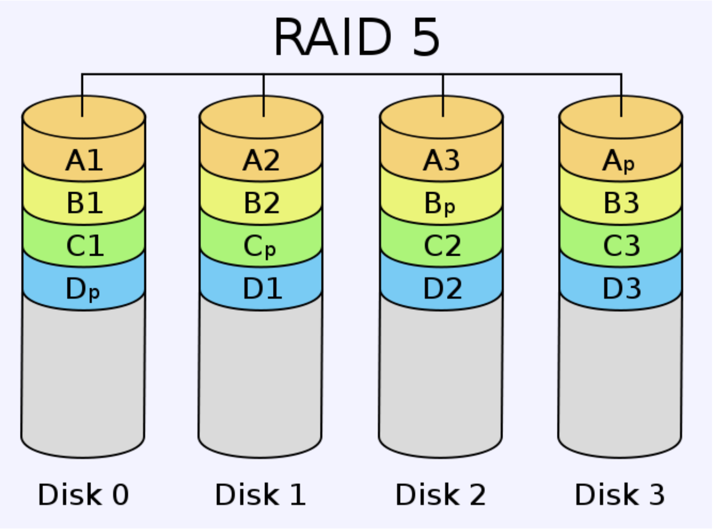
Drive Hot-Swapping in RAID

- In RAID a defect drive will be (ASAP)
  - manually swapped for a new drive (hot-swap), or
  - replaced by an idle drive (hot-spare) already in the system
- The new drive's content is rebuild by the RAID controller while the disk set is still operational
- Example RAID 4 with Disk 0 swapped:
  - $A1 = A2 \text{ EXOR } A3 \text{ EXOR } A_P$
  - $B1 = B2 \text{ EXOR } B3 \text{ EXOR } B_P$
  - $C1 = C2 \text{ EXOR } C3 \text{ EXOR } C_P$
  - $D1 = D2 \text{ EXOR } D3 \text{ EXOR } D_P$



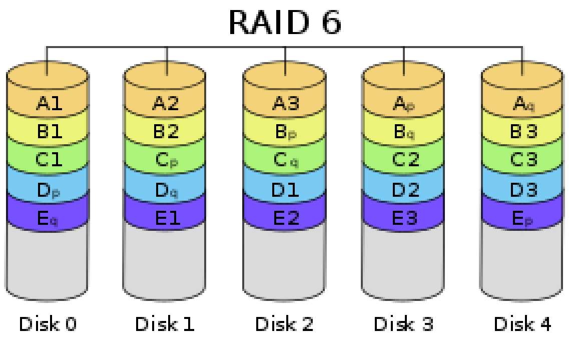
RAID 5

- Similar to RAID 4, but:
  - Block-level striping with distributed parity
  - Distributed parity evens out the stress of a dedicated parity disk among all RAID members
  - Write performance is increased since all RAID members participate in the serving of write requests
- Minimum of 3 drives required



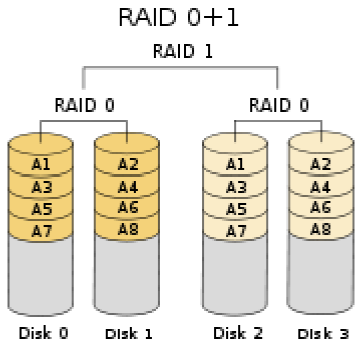
RAID 6

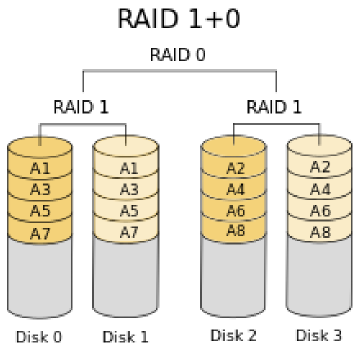
- RAID 6 extends RAID 5 by adding another parity block
  - thus, it uses block-level striping with two parity blocks distributed across all member disks
- Double catastrophic drive failures can be compensated (<u>any</u> two drives can fail)
- Minimum of 4 disks required
- Second parity involves EXOR function (as seen before) and a bit shift function



Other RAID Levels

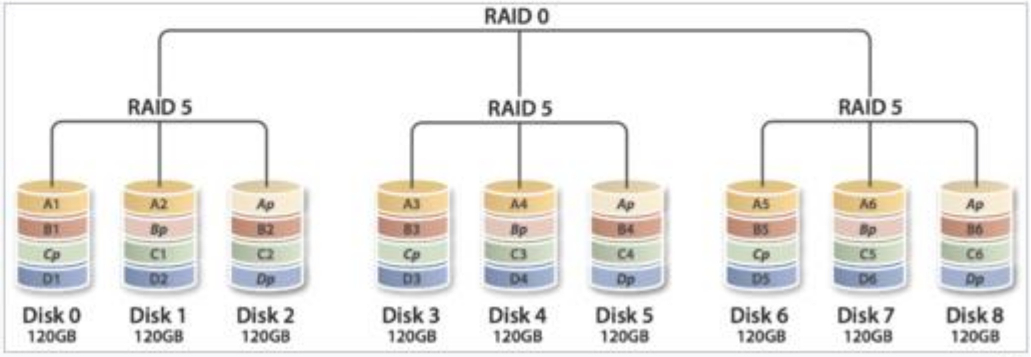
- RAID 0+1 (RAID 01)
  - 4 drives minimum
  - Some double catastrophic drive failures can be compensated
- RAID 1+0 (RAID 10)
  - 4 drives minimum
  - Some double catastrophic drive failures can be compensated
  - Best throughput (apart from RAID 0), so preferable RAID level for I/O-intensive applications

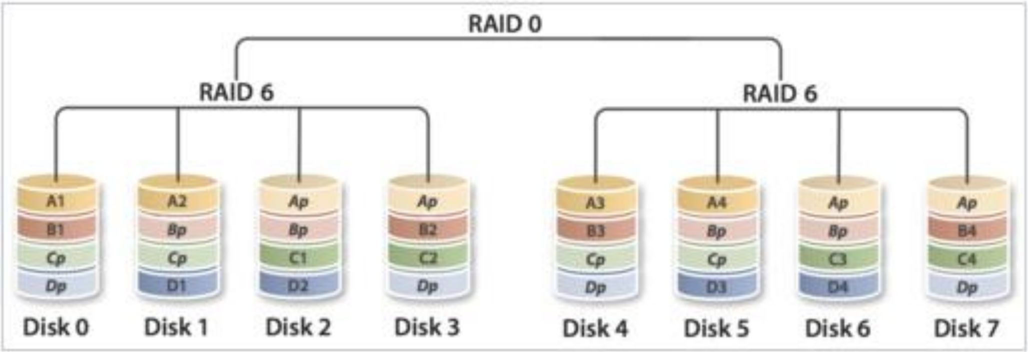




Other RAID Levels: RAID 5+0 (RAID 50) and RAID 6+0 (RAID 60)

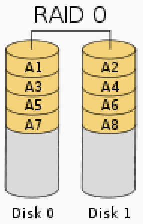
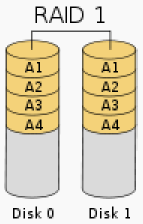
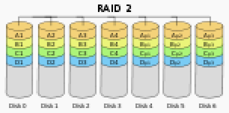
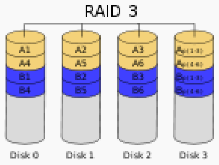
- □ Some triple catastrophic drive failures can be compensated
- □ Rebuild-time after drive swap reduced because of controller hierarchy
  - Faster turnaround time to restore full I/O capacity
  - Shorter vulnerable period where a second drive failure would be catastrophic



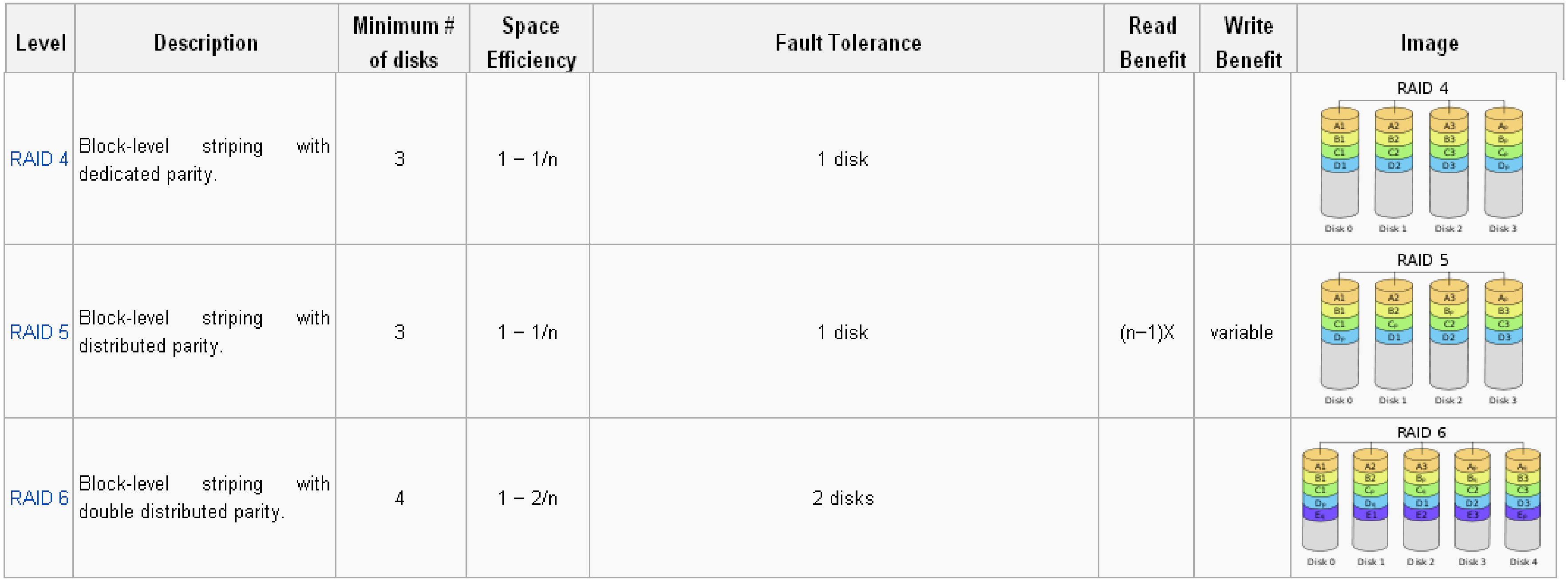


Mx2

Summary RAID 0 – RAID 3

Level	Description	Minimum # of disks	Space Efficiency	Fault Tolerance	Read Benefit	Write Benefit	Image
RAID 0	Block-level striping without parity or mirroring.	2	1	0 (none)	nX	nX	 RAID 0 A1 A3 A5 A7 A2 A4 A6 A8 Disk 0 Disk 1
RAID 1	Mirroring without parity or striping.	2	1/n	n−1 disks	nX	1X	 RAID 1 A1 A2 A3 A4 A1 A2 A3 A4 Disk 0 Disk 1
RAID 2	Bit-level striping with dedicated Hamming-code parity.	3	1 − 1/n · $\log_2(n-1)$	RAID 2 can recover from 1 disk failure or repair corrupt data or parity when a corrupted bit's corresponding data and parity are good.			 RAID 2 A1 B1 C1 D1 A2 B2 C2 D2 A3 B3 C3 D3 A4 B4 C4 D4 Ap1 Bp1 Cp1 Dp1 Ap2 Bp2 Cp2 Dp2 Ap3 Bp3 Cp3 Dp3 Disk 0 Disk 1 Disk 2 Disk 3 Disk 4 Disk 5 Disk 6
RAID 3	Byte-level striping with dedicated parity.	3	1 − 1/n	1 disk			 RAID 3 A1 A4 B1 B4 A2 A5 B2 B5 A3 A6 B3 B6 Ap(1-3) Ap(4-6) Bp(1-3) Bp(4-6) Disk 0 Disk 1 Disk 2 Disk 3

Summary RAID 4 – RAID 6



Level	Description	Minimum # of disks	Space Efficiency	Fault Tolerance	Read Benefit	Write Benefit	Image
RAID 4	Block-level striping with dedicated parity.	3	1 − 1/n	1 disk			RAID 4
RAID 5	Block-level striping with distributed parity.	3	1 − 1/n	1 disk	(n−1)X	variable	RAID 5
RAID 6	Block-level striping with double distributed parity.	4	1 − 2/n	2 disks			RAID 6

A1 B1 C1 D1
A2 B2 C2 D2
A3 B3 C3 D3
A_p B_p C_p D_p
Disk 0 Disk 1 Disk 2 Disk 3
A1 B1 C1 D_p
A2 B2 C_p D1
A3 B_p C2 D2
A_p B3 C3 D3
Disk 0 Disk 1 Disk 2 Disk 3
A1 B1 C1 D_p E_q
A2 B2 C_p D_q E1
A3 B_p C_q D1 E2
A_p B_q C2 D2 E3
A_q B3 C3 D3 E_p
Disk 0 Disk 1 Disk 2 Disk 3 Disk 4

GDPR

General Data Protection Regulation

- GDPR is a binding regulation in EU law on data protection in the EU and the European Economic Area (EEA), that became enforceable on 25 May 2018
- It also addresses the transfer of personal data outside the EU and EEA areas
- The GDPR's primary aim is to **enhance individuals' control and rights over their personal data and to simplify the regulatory environment for international business**
- The regulation **contains provisions and requirements related to the processing of personal data of individuals** who are located in the EEA, and applies to any enterprise—**regardless of its location and the data subjects' citizenship or residence**—that is processing the personal information of individuals inside the EEA

GDPR in Ireland

- GDPR applies to the majority of personal data processing tasks, but further rules on certain issues (for example the reasons for, and extent to which, data subject rights may be restricted) are set out in the **Data Protection Act 2018**
- Further on, the **Law Enforcement Directive** concerns the processing of personal data by law enforcement, i.e,. the prevention, investigation, detection or prosecution of criminal offences or the execution of criminal penalties

What is Data Protection?

- Data protection is about an **individual's fundamental right for privacy**
- When an individual gives their personal data to any organisation, the recipient has the duty to keep the data safe and private
- Data protection legislation
  - governs the way we deal with personal data / information
  - provides a mechanism for safeguarding privacy rights of individuals in relation to the processing of their data
  - upholds rights and enforces obligations

Recall GDPR: Personal Data

- Any information relating to an identified or identifiable natural person ('data subject’)
- An identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person"

Recall GDPR: Sensitive Personal Data

- This includes
  - Racial origin
  - Political opinions
  - Religious or philosophical beliefs
  - Trade Union membership
  - Genetic data (e.g. biological samples)
  - Biometric data (e.g. fingerprints)
  - Data concerning health
  - Data concerning a person's sex life or sexual orientation

- GDPR requires explicit consent to process special categories of personal data

Recap Cookies

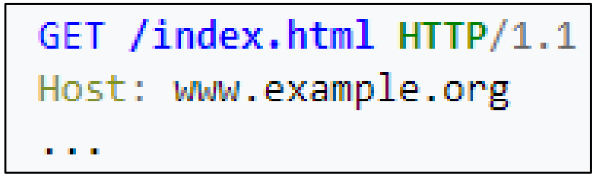
- An (HTTP) cookie is a small piece of data stored on the user's computer by the web browser while browsing a website
- Cookies were designed to be a reliable mechanism for websites to remember stateful information (such as items added in the shopping cart in an online store) or to record the user's browsing activity
- They can also be used to remember pieces of information that the user previously entered into form fields
- Authentication cookies are the most common method used by web servers to know whether the user is logged in or not, and which account they are logged in with

Cookie Implementation

- Cookies are arbitrary pieces of data (i.e. large random strings), usually chosen and first sent by the web server, and stored on the client computer by the web browser
- The browser then sends them back to the server with every request
- Browsers are required to:
  - support cookies as large as 4,096 bytes in size
  - support at least 50 cookies per domain (i.e. per website)
  - support at least 3,000 cookies in total

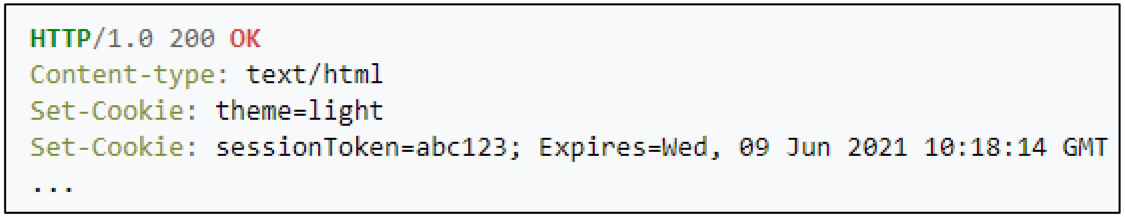
Setting a Cookie - Example

- A browser sends its first request for the homepage of www.example.org, resulting in the GET request



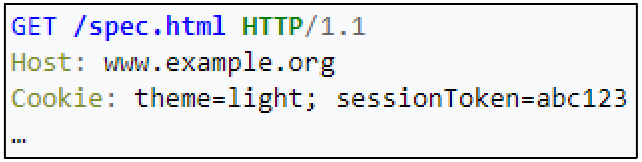
```
GET /index.html HTTP/1.1
Host: www.example.org
...
```

- The server responds with



```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: theme=light
Set-Cookie: sessionToken=abc123; Expires=Wed, 09 Jun 2021 10:18:14 GMT
...
```

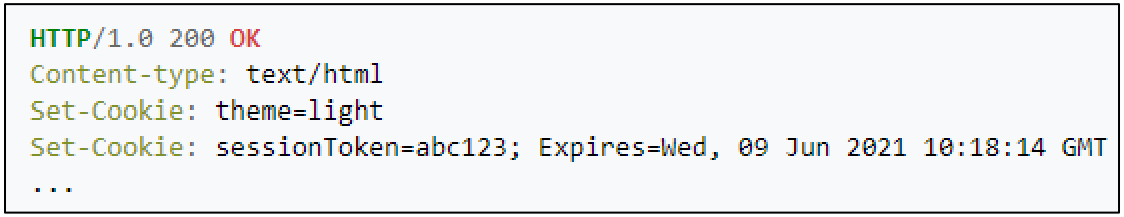
- Later client requests to this server will contain these cookies:



```
GET /spec.html HTTP/1.1
Host: www.example.org
Cookie: theme=light; sessionToken=abc123
…
```

Cookie Structure

- A cookie consists of the following components:
  - Name
  - Value
  - Zero or more attributes (name/value pairs) Attributes store information such as the cookie's expiration, domain, and flags (such as *Secure* and *HttpOnly*)



```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: theme=light
Set-Cookie: sessionToken=abc123; Expires=Wed, 09 Jun 2021 10:18:14 GMT
...
```

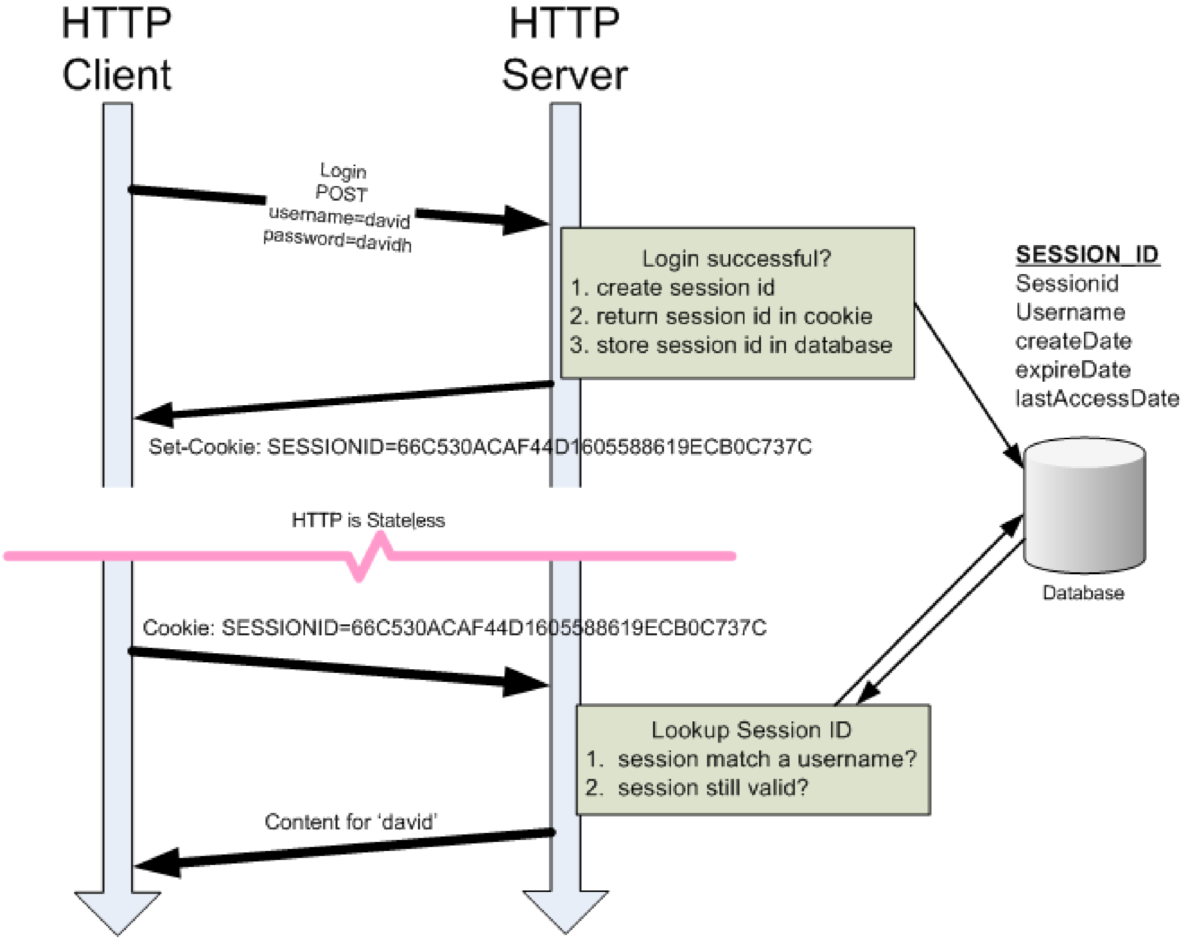
Session Cookies

- A session cookie (aka in-memory cookie, transient cookie or non-persistent cookie) exists only in temporary memory while the user navigates its website
- Web browsers normally delete session cookies when the user closes the browser
- Session cookies do not have an expiration date assigned to them, which is how the browser knows to treat them as session cookies
- Example: “theme” cookie on previous slide

Persistent Cookie

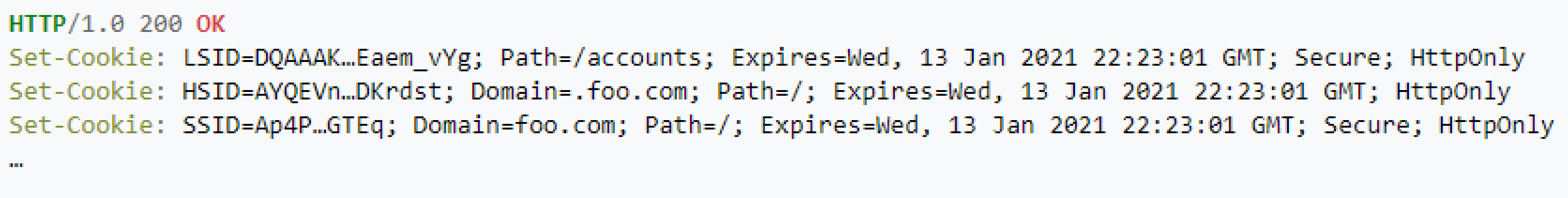
- A persistent cookie expires at a specific date or after a specific length of time
- For the persistent cookie's lifespan set by its creator, its information will be transmitted to the server every time the user visits the website that it belongs to
- … or every time the user views a resource belonging to that website from another website (such as an advertisement).
  For this reason, persistent cookies are sometimes referred to as tracking cookies because they can be used by advertisers to record information about a user's web browsing habits
- However, they are mainly used for legitimate reasons, such as keeping users logged into their accounts on websites, to avoid re-entering login credentials at every visit
- Example: "sessionToken" cookie in the previous example

Session Management via Persistent Cookies



Cookie Attributes

- Consider the following response header sent by a webserver that contains 3 persistent cookies:

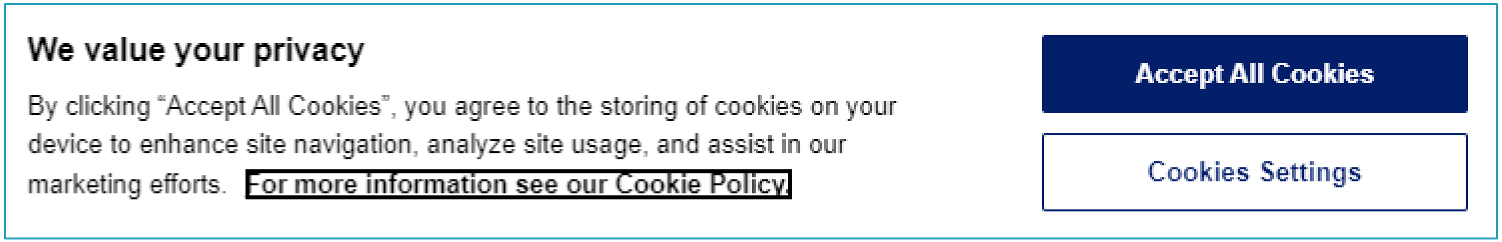


```
HTTP/1.0 200 OK
Set-Cookie: LSID=DQAAAK…Eaem_vYg; Path=/accounts; Expires=Wed, 13 Jan 2021 22:23:01 GMT; Secure; HttpOnly
Set-Cookie: HSID=AYQEVn…DKrdst; Domain=.foo.com; Path=/; Expires=Wed, 13 Jan 2021 22:23:01 GMT; HttpOnly
Set-Cookie: SSID=Ap4P…GTEq; Domain=foo.com; Path=/; Expires=Wed, 13 Jan 2021 22:23:01 GMT; Secure; HttpOnly
…
```

- The *Domain* and *Path* attributes define the cookie's scope
- The *Secure* attribute makes sure that the cookie can only be transmitted over an encrypted connection (i.e. HTTPS → later), making it a **secure cookie**
- The *HttpOnly* attribute directs browsers not to expose cookies through channels other than HTTP / HTTPS requests
  This means that this **HttpOnly cookie** cannot be accessed via client-side scripting languages (notably JavaScript)

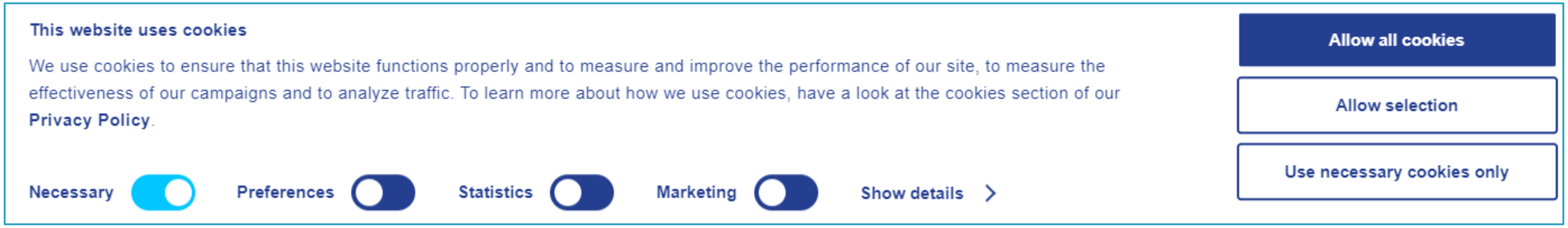
GDPR and Cookies

- Generally, a user's consent must be sought before a cookie is installed in a web browser



We value your privacy

By clicking "Accept All Cookies", you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. For more information see our Cookie Policy



This website uses cookies

We use cookies to ensure that this website functions properly and to measure and improve the performance of our site, to measure the effectiveness of our campaigns and to analyze traffic. To learn more about how we use cookies, have a look at the cookies section of our **Privacy Policy**.

Necessary ● Preferences ○ Statistics ○ Marketing ○

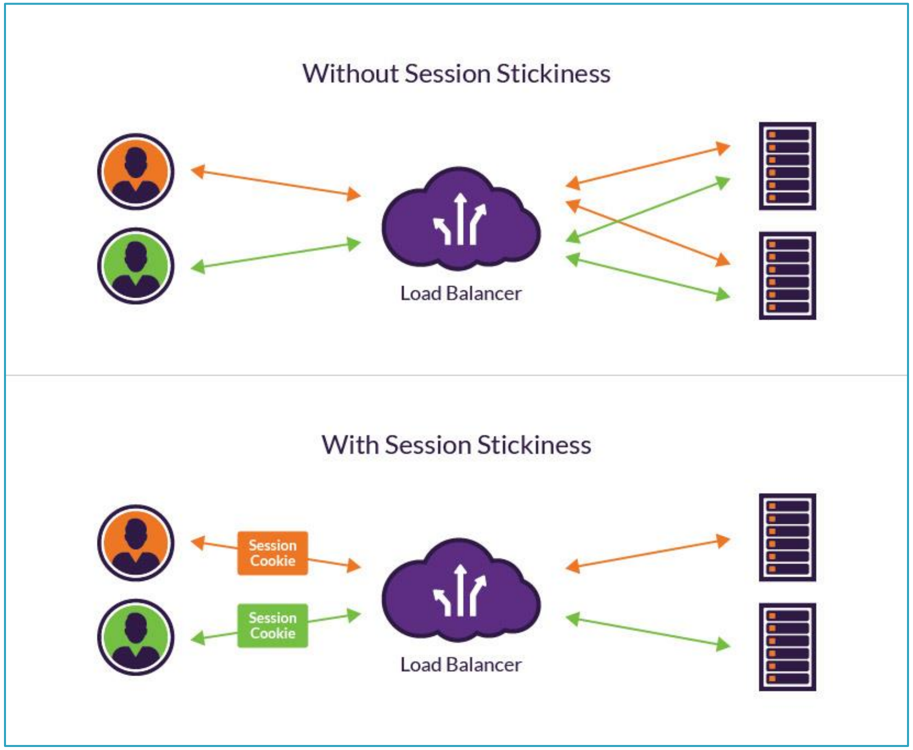
-
  - The communications exemption
  - The strictly necessary exemption

The Communications Exemption

- This applies to cookies whose sole purpose is for carrying out the transmission of a communication over a network, for example to identify the communication endpoints
- Example: load-balancing cookies that distribute network traffic across different backend servers, aka **session stickiness**
  - Here a load balancer **creates an affinity** between a client and a specific network server for the duration of a session using a cookie with a random and unique tracking id
  - Subsequently, for the duration of the session, the load balancer routes all of the requests of this client to a specific backend server using the tracking id

Session Stickiness

- Top image:
  - No load balancing at all
- Bottom image:
  - The LB generates and returns a tracking cookie back to a client when its session is initiated
  - This cookie is tagged to every subsequent client request and allows the LB to forward the request to always the same server (therefore the stickiness)



The *strictly necessary Exemption*

- Must be linked to a service delivered over the internet, i.e. a website or an app
- This service must have been explicitly requested by the user (i.e. typing in the URL) and the use of the cookie must be restricted to what is strictly necessary to provide that service
- Note that cookies related to advertising are not strictly necessary and must be consented to

Example for the *strictly necessary Exemption*

- A website uses session cookies to keep track of items a user places in an online shopping basket
  - Assuming this cookie will be deleted once the session is over
- Cookies that record a user's language or country preference

Data Processing

- Performing any operation on personal data, manually or by automate means, including:
  - Obtaining
  - Storing
  - Transmitting
  - Recording
  - Organising
  - Altering
  - Disclosing
  - Erasing

Entities in GDPR

- GDPR distinguishes between:
  - The Data Subject
  - The Data Protection Officer (DPO)
  - The Data Controller
  - The Data Processor

The Data Subject

- This is the (living!) person to whom the data relates
- Under GDPR, businesses have a legal obligation to keep their data up-to-date, which means that, theoretically, data of deceased should be removed
- Deceased persons, who predate the introduction of GDPR, may be covered by national legislation active at the time of death (e.g. the Data Protection Acts 1988 and 2003 in Ireland)
- Access by a next-of-kin to the personal data (e.g. emails) of a deceased person may be possible under Irish **Freedom of Information laws**

The Data Protection Officer (DPO)

- The primary role of the DPO is to ensure that her organisation processes the personal data of its staff, customers, and other data subjects in compliance with the applicable data protection rules
- It is a mandatory role within three different scenarios:
  - When the processing is undertaken by a public authority or body
  - When an organisation's main activities require the frequent and large-scale monitoring of individual people
  - Where large scale processing of special categories of data or data relating to criminal records forms the core activities
- The Data Protection Officer is required to be an expert within this field, along with the requirement for them to report to the highest management level.
  - With this being a challenging aspect of GDPR compliance for smaller organisations, there is the option to make an external appointment of a third-party

The Data Controller

- The Data Controller is the company or an individual who has overall control over the processing of personal data
- The Data Controller takes on the responsibility for GDPR compliance
- A Data Controller needs to have had sufficient training and be able to competently ensure the security and protection of data held within the organisation

The Data Processor

- The Data Processor is the person who is responsible for the processing of personal information
- Generally, this role is undertaken under the instruction of the data controller
    - This might mean obtaining or recording the data, it's adaption and use. It may also include the disclosure of the data or making it available for others
- Generally, the Data Processor is involved in the more technical elements of the operation, while the interpretation and main decision making is the role of the Data Controllers

Cloud Services and GDPR

- A Cloud Service Provider will be considered a **Data Processor** under GDPR if it provides data processing services (e.g. storage) on behalf of the Data Controller even without determining the purposes and means of processing
- A Cloud Service Provider that offers personal data processing services directly to Data Subjects will be **Data Controller**

Some Key Benefits for Data Subjects

- More information must be given to data subjects (e.g. how long data will be kept, right to lodge a complaint)
- Must explain and document legal basis for processing personal data
- Tightens the rules on how consent is obtained (must be distinguishable from other matters and in clear plain language)
- Must be as easy to withdraw consent as it is to give it
- Mandatory notification of security breaches without undue delay
  - To data protection commissioner within 72 hours

Personal Data Security Breaches

- Disclosure of confidential data to unauthorised individuals
- Loss or theft of data or equipment on which data is stored
- Hacking, viruses or other security attacks on IT equipment/ systems / networks
- Inappropriate access controls allowing unauthorised use of information
- Emails containing personal data sent in error to wrong recipient
- Applies to paper and electronic records

Some Key Benefits for Data Subjects

- Right of Access (copy to be provided within one month)
- Right to erasure (i.e. right to be forgotten)
- Right to restriction of processing
- Right to object to processing
- Right not to be subject to a decision based solely on automated processing

GDPR Key Principles

- The GDPR sets out several key principles:
  1. Lawfulness
  2. Fairness and transparency
  3. Purpose limitation
  4. Data minimisation
  5. Accuracy
  6. Storage limitation
  7. Integrity and confidentiality (security)
  8. Accountability

GDPR Principle: Lawfulness

- You must **identify valid grounds** under the GDPR (known as a ‘lawful basis’) for collecting and using personal data
- Processing shall be lawful only if and to the extent that at least one of the following applies:
  - Consent
  - Necessary for the performance of a contract
  - Necessary for compliance with a legal obligation
  - Necessary to protect the vital interests of the data subject or another person
  - Necessary for the performance of a task carried out in the public interest
  - Necessary for the purpose of the legitimate interests

GDPR Principle: Fairness and Transparency

- You must **use personal data in a way that is fair**; this means you must not process the data in a way that is unduly detrimental, unexpected or misleading to the individuals concerned
- You must be **clear, open and honest with people** from the start about how you will use their personal data
- At the time personal data is being collected from data subjects, they must be informed via a "Data Protection Notice"

Data Protection Notice

- A data protection notice entails the following:
  - The identity and contact details of the data controller
  - The contact details of the data protection officer
  - The purpose of the processing and the legal basis for the processing
  - The recipients or categories of recipients of the data
  - Details of any transfers out of the EEA, safeguards in place and the means by which to obtain a copy of them
  - The data retention period used or criteria to determine same
  - The individual's rights (access, rectification and erasure, restriction, complaint)

GDPR Principle: Purpose Limitation

- You must be **clear about what your purposes** for processing are from the start
- You need to **record your purposes** as part of your documentation obligations and specify them in your privacy information for individuals
- You **can only use the personal data for a new purpose** if either this is compatible with your original purpose, you get consent, or you have a clear basis in law

GDPR Principle: Data Minimisation

- You must ensure the personal data you are processing is:
  - **adequate** – sufficient to properly fulfil your stated purpose
  - **relevant** – has a rational link to that purpose
  - **limited** to what is necessary – you do not hold more than you need for that purpose

GDPR Principle: Accuracy

- You should take all reasonable steps to ensure the personal data you hold **is not incorrect or misleading** as to any matter of fact
- You may need to **keep the personal data updated,** although this will depend on what you are using it for
- If you **discover that personal data is incorrect or misleading**, you must take reasonable steps to correct or erase it as soon as possible
- You must **carefully consider any challenges to the accuracy** of personal data

GDPR Principle: Storage Limitation

- You must not keep personal data **for longer than you need it**
- You need to think about – and be able to justify – **how long you keep personal data**; this will depend on your purposes for holding the data
- You need a policy **setting standard retention periods** wherever possible, to comply with documentation requirements
- You should also **periodically review the data you hold**, and erase or anonymise it when you no longer need it
- You must **carefully consider any challenges to your retention of data**; individuals have a right to erasure if you no longer need the data
- You can **keep personal data for longer if you are only** keeping it for public interest archiving, scientific or historical research, or statistical purposes

GDPR Principle: Accountability and Governance

- Accountability is one of the data protection principles - it makes you responsible for complying with the GDPR and says that **you must be able to demonstrate your compliance**
- You need to put in place appropriate technical and organisational measures to meet the requirements of accountability

GDPR Principle: Accountability and Governance

- Accountability requires controllers to maintain records of processing activities in order to demonstrate how they comply with the data protection principles, i.e.
  - Inventory of personal data
  - Providing assurance about compliance
  - Need to document
    - Why it is held
    - How it is collected
    - When it will be deleted
    - Who may gain access to it

GDPR Principle: Integrity and Confidentiality

- A key principle of the GDPR is that you process personal data securely by means of 'appropriate technical and organisational measures' – this is the 'security principle'
- Doing this requires you to consider things like risk analysis, organisational policies, and physical and technical measures
- Where appropriate, you should look to use measures such as **pseudonymisation and encryption**
- Your measures must ensure the '**confidentiality, integrity and availability**' of your systems and services and the personal data you process within them
- The measures must also enable you to **restore access and availability** to personal data in a timely manner in the event of a physical or technical incident

CT437 COMPUTER SECURITY AND FORENSIC COMPUTING

HISTORY OF CRYPTOGRAPHY
CRYPTOGRAPHIC CONCEPTS

Dr. Michael Schukat

OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Lecture Outline and Motivation

- Recap: Technologies used to ensure confidentiality:
  - Encryption (obviously)
  - Access Control (e.g. multi-factor authentication)
  - Secure network protocols
- Therefore, this lecture provides:
  - A summary of terms linked to cryptography
  - An overview of historic cryptographic algorithms (recap CT255)
  - Some context for the next topic, **modern cryptography**

PASSWORD
******
MULTIFACTOR AUTHENTICATION
LOGGED IN
NIST National Institute of Standards and Technology U.S. Department of Commerce
www.nist.gov

Basic Terminology

- Cryptography
  - The art of encompassing the principles and methods of transforming an intelligible message into one that is unintelligible, and then retransforming that message back to its original form
    - Intelligible means “able to be understood” or comprehensible

Basic Terminology

- Plaintext
  - The original intelligible message, e.g. “THIS IS A SECRET MESSAGE”
- Ciphertext
  - The transformed message, e.g. “XPHDSYUEGSD68G4AS8AG56”
- Cipher
  - An algorithm for transforming an intelligible message into one that is unintelligible
- Key
  - Some critical information used by the cipher, known only to the sender & receiver
  - Selected from a **keyspace** K (i.e., a set of all possible keys)

Basic Terminology

- Encipher (encode)
  - The process of converting plaintext to ciphertext using a cipher and a key
- Encryption
  - The mathematical function $E_K()$ mapping plaintext P to ciphertext using the specified key K:

    $$C = E_K(P)$$

Basic Terminology

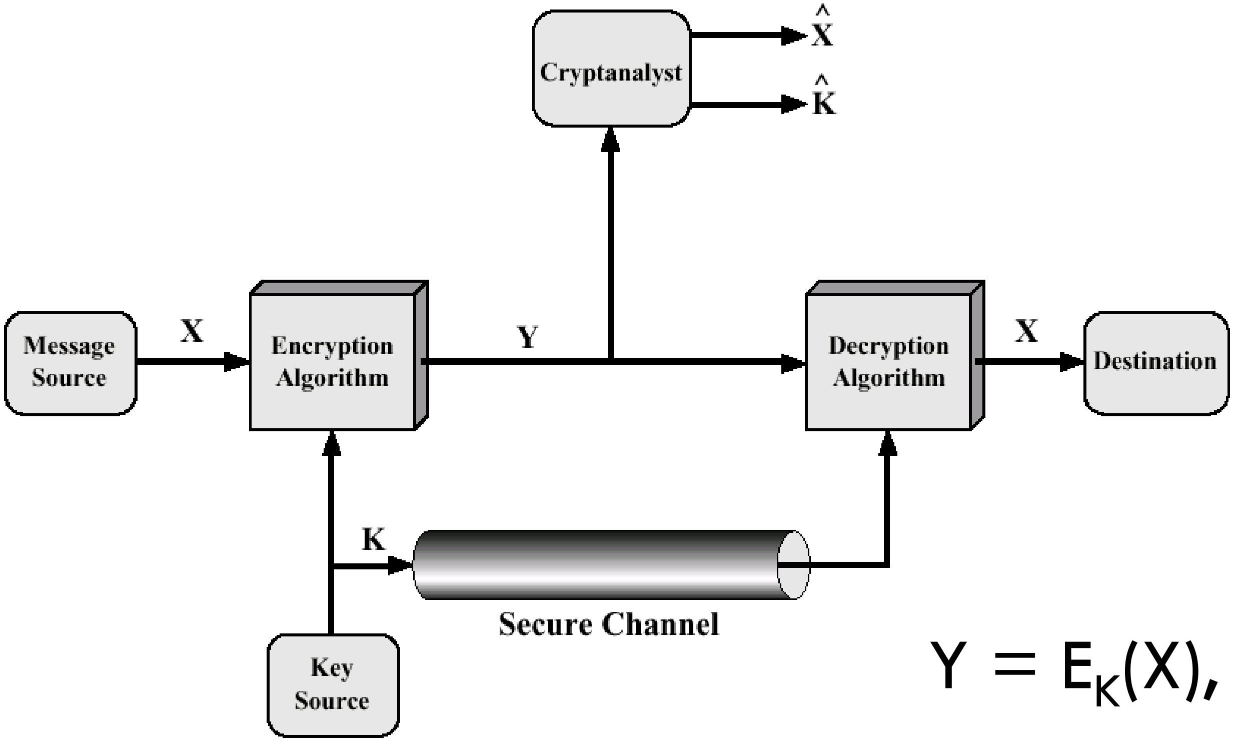
- Decipher (decode)
  - The process of converting ciphertext back into plaintext using a cipher and a key
- Decryption:
  - The mathematical function $E_K^{-1}()$ mapping ciphertext C to plaintext P using the specified key K:

    $$P = E_K^{-1}(C)$$

Basic Terminology

- Cryptanalysis
  - The study of principles and methods of transforming an unintelligible message back into an intelligible message without knowledge of the key
- Cryptology
  - The field encompassing both cryptography and cryptanalysis

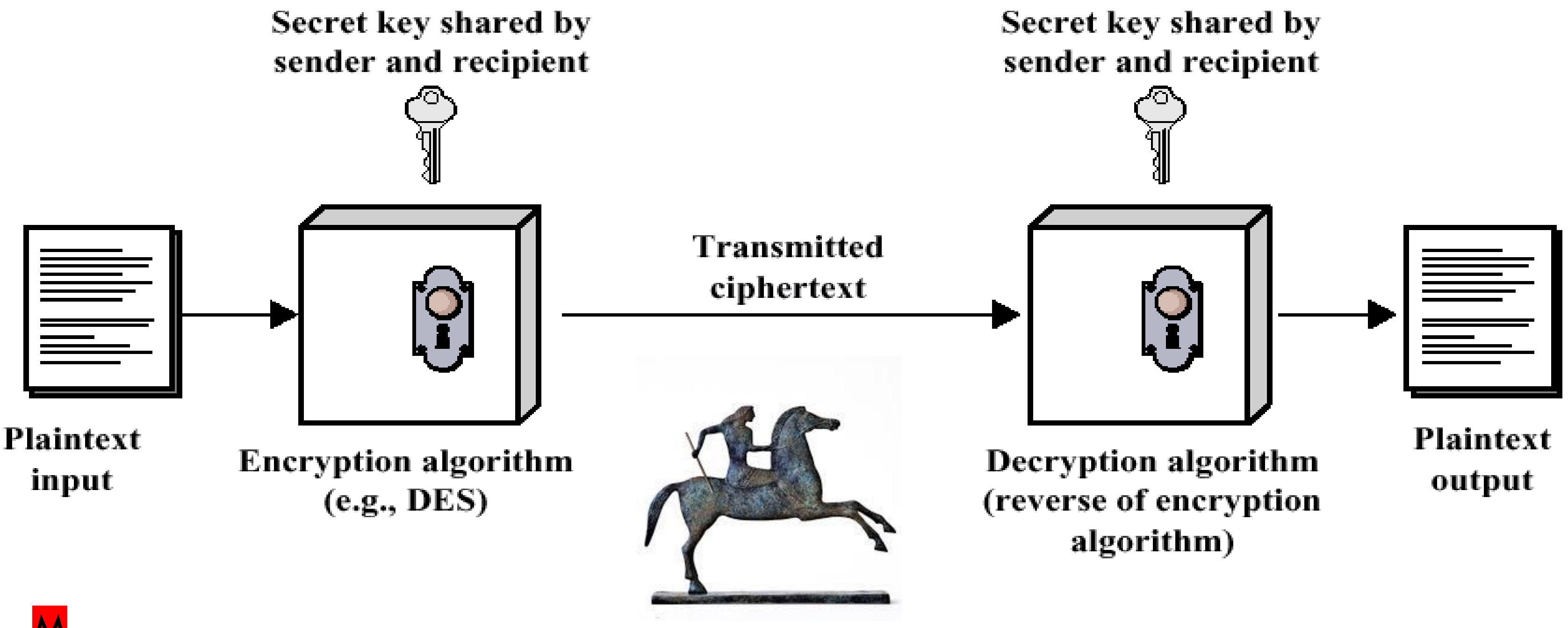
Model of Conventional Cryptosystem



$$Y = E_K(X),\ X = E_K^{-1}(Y)$$

Classical Cryptography

- Ancient ciphers have been in use for over 5,000 years
- Already used by ancient Egyptians, Hebrews and Greeks
- Normally they would follow the following scheme:



Secret key shared by sender and recipient

Secret key shared by sender and recipient

Transmitted ciphertext

Plaintext input

Encryption algorithm (e.g., DES)

Decryption algorithm (reverse of encryption algorithm)

Plaintext output

Caesar Cipher

- 2000 years ago, Julius Caesar used a simple substitution cipher, now known as the Caesar cipher
- First attested use in military affairs (Gallic Wars)
- Replace each letter by 3rd letter on, e.g.

  L FDPH L VDZ L FRQTXHUHG ->
  I CAME I SAW I CONQUERED

- We can describe this mapping (or translation alphabet) as:

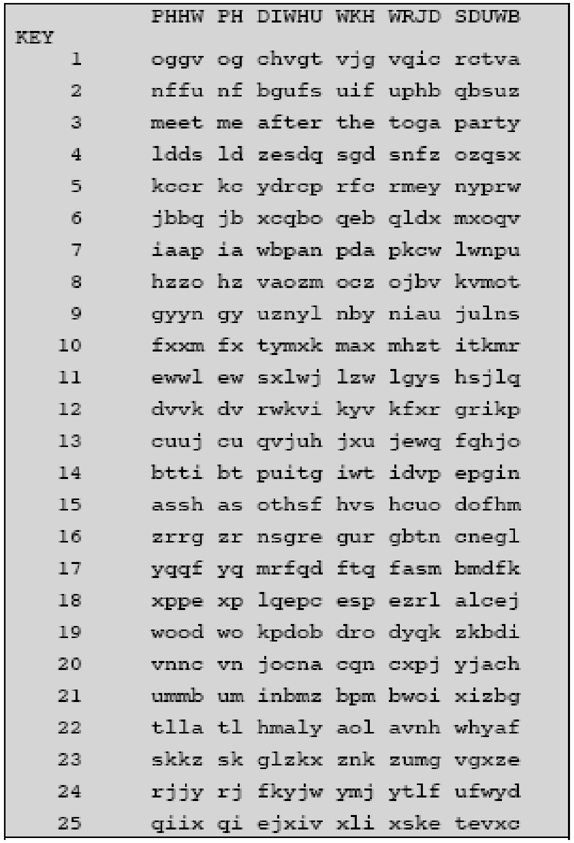
  Plain: ABCDEFGHIJKLMNOPQRSTUVWXYZ
  Cipher: DEFGHIJKLMNOPQRSTUVWXYZABC

Generalised Caesar Cipher

- More generally can use any shift from 1 to 25, i.e. replace each letter of message by a letter a fixed distance away
- Specify key letter as the letter a plaintext A maps to,
  - e.g. a key letter of F means
    A maps to F, B to G, … Y to D, Z to E
    e.g. shift letters by 5 places
- Hence have 26 (25 useful) ciphers
- Note that with this and all other historic ciphers punctuation and spaces are ignored and all text is converted to capital letters

How to break the generalised Caesar Cipher

- Try all 25 possibilities until you recover some meaningful text



```
             PHHW PH DIWHU WKH WRJD SDUWB
KEY
    1        oggv og chvgt vjg vqic rctva
    2        nffu nf bgufs uif uphb qbsuz
    3        meet me after the toga party
    4        ldds ld zesdq sgd snfz ozqsx
    5        kccr kc ydrcp rfc rmey nyprw
    6        jbbq jb xcqbo qeb qldx mxoqv
    7        iaap ia wbpan pda pkcw lwnpu
    8        hzzo hz vaozm ocz ojbv kvmot
    9        gyyn gy uznyl nby niau julns
   10        fxxm fx tymxk max mhzt itkmr
   11        ewwl ew sxlwj lzw lgys hsjlq
   12        dvvk dv rwkvi kyv kfxr grikp
   13        cuuj cu qvjuh jxu jewq fqhjo
   14        btti bt puitg iwt idvp epgin
   15        assh as othsf hvs hcuo dofhm
   16        zrrg zr nsgre gur gbtn cnegl
   17        yqqf yq mrfqd ftq fasm bmdfk
   18        xppe xp lqepc esp ezrl alcej
   19        wood wo kpdob dro dyqk zkbdi
   20        vnnc vn jocna cqn cxpj yjach
   21        ummb um inbmz bpm bwoi xizbg
   22        tlla tl hmaly aol avnh whyaf
   23        skkz sk glzkx znk zumg vgxze
   24        rjjy rj fkyjw ymj ytlf ufwyd
   25        qiix qi ejxiv xli xske tevxc
```

The Mono-Alphabetic (or simple) Substitution Cipher

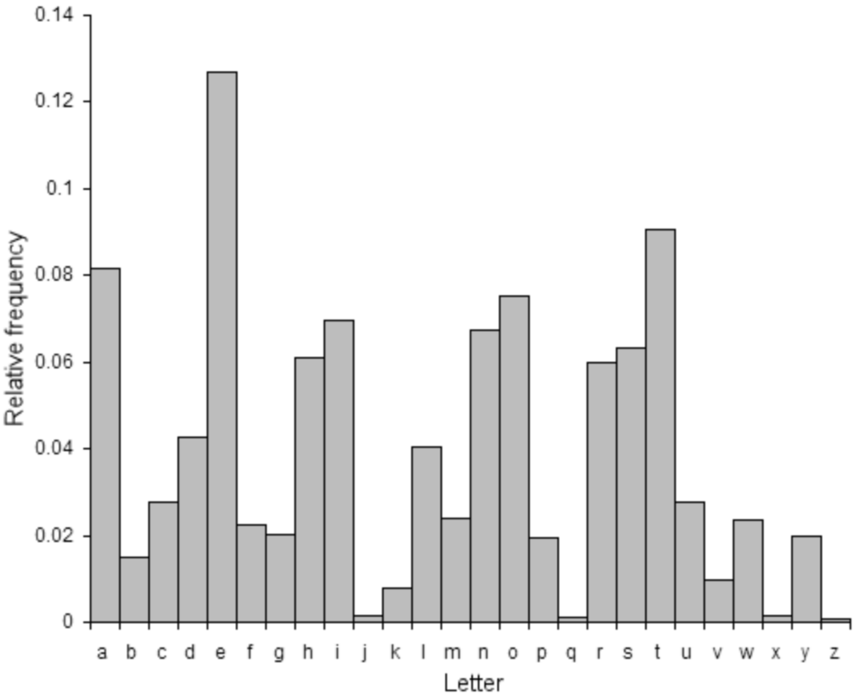
- In the mono-alphabetic (or simple substitution) cipher each letter of the plain text is replaced with another letter of the alphabet
- It uses a fixed key which consist of the 26 letters of a “shuffled alphabet”.
- Example:
  - Plaintext alphabet (obviously): ABCDEFGHIJKLMNOPQRSTUVWXYZ
  - Ciphertext alphabet (i.e. the key): ZEBRASCDFGHIJKLMNOPQTUVWXY
  - Plaintext message: FLEEATONCEWEAREDISCOVERED
  - Ciphertext message: SIAAZQLKBAVAZOARFPBLUAOAR
- This ciphers allows for 26! (= 4.0329146e+26) possible key combinations …
  - This is too many combinations for a brute force attack where the attacker tries every single possible key!
    - This of course assumes that the attacker can identify the correctly decoded cyphertext (e.g., a text written in English)
- **Is this cipher therefore unbreakable?**

Non-trivial Cryptanalysis Attacks Against Substitution Ciphers

- Frequency Analysis:
  This attack relies on the fact that certain letters or symbols occur more frequently in the English language than others. By analysing the frequency of characters in the ciphertext, one can make educated guesses about the substitutions made in the cipher, ultimately revealing the plaintext
  See also next slides
- Pattern Recognition:
  Cryptanalysts may also exploit patterns in the ciphertext to deduce information about the key. Recognisable patterns, such as common word endings or repeating character sequences, can provide valuable clues about the substitutions used in the cipher
- Known-Plaintext Attack:
  See next slides

Cryptanalysis via Letter Frequency Analysis

- In most written languages, letters are not equally commonly used
- For example, in the English language:
  - E is by far the most common letter followed by T,R,N,I,O,A,S
  - Other letters like Z,J,K,Q,X are fairly rare
  - See frequency table on the right
- There are tables of single, double & triple letter frequencies for all common languages
- There is an example for the cryptanalysis of a ciphertext via letter frequency analysis at the end of this slide deck



C-Program for Letter Frequency Analysis

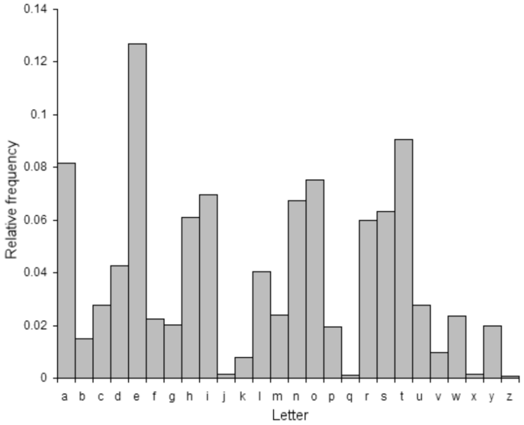
```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main(int argc, char *argv[])
{
    FILE *fp;
    int data[26];
    char c;
    int i;

    memset(data, 0, sizeof(data));

     if (argc != 2)
       return(-1);

    if ((fp = fopen(argv[1], "r")) == NULL)
            return(-2);

    while (!feof(fp))
    {
            c = toupper(fgetc(fp));

            if ((c >= 'A') && (c <= 'Z'))
                        data[c - 65]++;
    }

    for (i = 0; i < 26; i++)
            printf("%c: %i\n", i + 65, data[i]);

    fclose(fp);
    return(1);
}
```

Known Plaintext Attacks

- The known-plaintext attack (KPA) is an attack model for cryptanalysis where the attacker has access to both
  - (some of) the plaintext (this is called a crib),
  - and its encrypted version
- See the example on the next slide

Example: Combined known-Plaintext and Pattern Recognition Attack

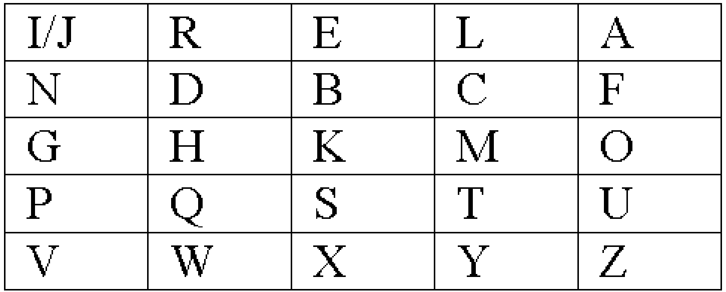
- You are presented with the following ciphertext which is based on a substitution cipher:
  JEPOUMJWFIFSFCVUNZIPNFJTNZDBTUMFGVMMTUPQ
- You know the original plaintext message consists of capital letters only (no spaces) and contains the following plaintext **crib**:
  MYHOMEISMYCASTLE
- How could you tackle this?



Playfair Cipher

- Not even the large number of keys in a monoalphabetic substitution cipher provides security!
- One approach to improving security was to encrypt multiple letters at once
- The **Playfair Cipher** is an example for such an approach
- Algorithm was invented by Charles Wheatstone in 1854, but named after his friend Baron Playfair

Playfair Cipher



I/J	R	E	L	A
N	D	B	C	F
G	H	K	M	O
P	Q	S	T	U
V	W	X	Y	Z

- How it works:
  - Create a 5x5 grid of letters; insert the keyword as shown, with each letter only considered once; fill the grid with the remaining letters in alphabetic order
  - Letters I and J are treated the same (see example above with keyword IRELAND)
  - Letters are encrypted in pairs
  - Repeats have an X inserted:
    BALLOON -> BA LX LO ON
  - Letters that fall in the same row are each replaced with the letter on the right (OK becomes GM)
  - Letters in the same column are replaced with the letter below (FO becomes OU)
  - Otherwise, each letter gets replaced by the letter in its row but in the other letters column (QM becomes TH)

Robustness of Playfair Cipher

- The algorithm' complexity was much improved over the simple monoalphabetic cipher, since we have 26 x 26 (= 676) character combinations we have to deal with
- This requires a 676-entry frequency table for analysis (versus 26 for a monoalphabetic cipher) and substantially more ciphertext for a cryptanalysis
- Therefore, it was widely used for many years, e.g., by US & British military in WW1
- However, it **can** be broken, given a few hundred letters

Example Playfair Cipher

- Consider the Playfair Cipher and the key "PRUNEJUICE"
- Encipher the following plaintext: "KENSENTMEX"
- What is the resulting ciphertext?

Vigenère Cipher

- Blaise de Vigenère is generally credited as the inventor of the "polyalphabetic substitution cipher“
  - A monoalphabetic cipher is any cipher in which the letters of the plain text are mapped to cipher text letters based on a single alphabetic key
  - A polyalphabetic substitution ciphers uses multiple substitution alphabets
- To improve security, use many monoalphabetic substitution alphabets
- Hence each letter can be replaced by many others
- Use a key to select which alphabet is used for each letter of the message
- ith letter of key specifies ith alphabet to use
- Use each alphabet in turn
- Repeat from start after end of key is reached

Example Vigenère Cipher

- Write the plaintext out and under it write the keyword repeated
- Then using each key letter in turn as a Caesar cipher key
- Encrypt the corresponding plaintext letter. Example:

Plaintext THISPROCESSCANALSOBEEXPRESSED

Keyword CIPHERCIPHERCIPHERCIPHERCIPHE

Ciphertext VPXZTIQKTZWTCVPSWFDMTETIGAHLH

In this example have the keyword "CIPHER". Hence have the following translation alphabets:

C -> CDEFGHIJKLMNOPQRSTUVWXYZAB

I -> IJKLMNOPQRSTUVWXYZABCDEFGH

... ...

ABCDEFGHIJKLMNOPQRSTUVWXYZ

to map the above plaintext letters

Example Vigenère Cipher

- Encode the plaintext “**KENSENTME**” using the Vigenère cipher and the keyword “BABA”

Plaintext:	KENSENTME
Key:	BABABABAB
Ciphertext:	MFPTGOVNG

How to break the Vigenère Cipher

- Search the ciphertext for repeated strings of letters; the longer strings you find the better
- For each occurrence of a repeated string, count how many letters are between the first letters in the string and add one
- Factor the number you got in the above computation (e.g., 2, 5 and 10 itself are factors of 10)
- Repeat this process with each repeated string you find and make a table of common factors. The most common factor is probably the length of the keyword that was used to encipher the ciphertext. Call this number 'n'
- Do a frequency count on the ciphertext, on every nth letter. You should end up with n different frequency counts
- Compare these counts to standard frequency tables to figure out how much each letter was shifted by
- Undo the shifts and read off the message!

Example Breaking the Vigenère Cipher

Key: ABCDABCDABCDABCDABCDABCDABCD (not known to attacker)
Plaintext: **CRYPTO**ISSHORTFOR**CRYPTO**GRAPHY (not known to attacker)
Ciphertext: CSASTPKVSIQUTGQUCSASTPIUAQJB

- Our search reveals to following repetition:
  CSASTP KV SIQUT GQU **CSASTP**IUAQJB

- The distance is 16, therefore the key length n is either 2, 4, 8 or 16 characters
- Do four different frequency counts on the ciphertext, i.e., on every nth letter
- Continue as shown before

In-Class Activity: Breaking Vigenère

- Consider the following ciphertext that has been encoded using a Vigenère Cipher:

DYDUXRMHTVDVNQDQNWDYDUXRMHARTJGWNQD

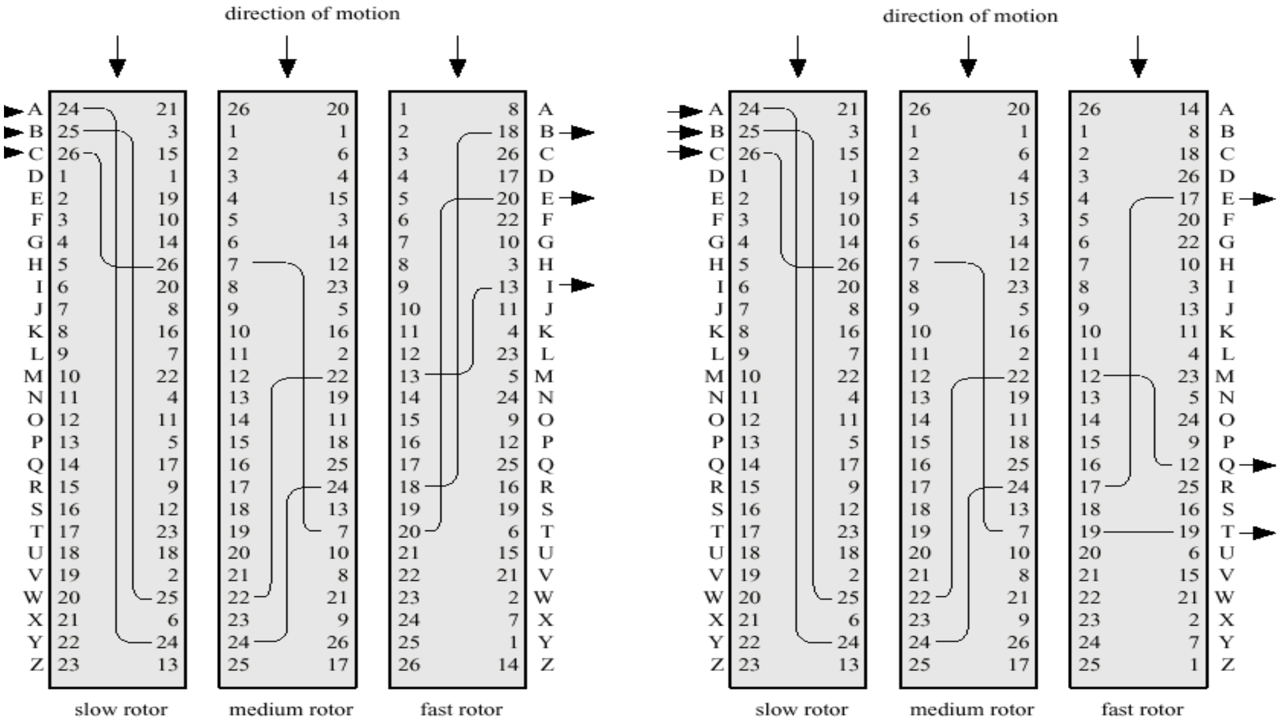
- Q1: Which repeating strings can you identify?
- Q2: What is the distance of their appearances?
- Q3: Subsequently, what is the probable key length?

Rotor Machines

- These allowed for the mechanisation / automation of message encryption and decryption and were widely used in the 20th century (until the 1970s)
- The primary components of a rotor machine are
  - a set of rotors
  - a keyboard for inputting text
  - A dashboard (e.g. array of letter-coded lamps) to show the output
- Rotors are rotating disks with an array of electrical contacts on either side
- The wiring between the contacts implements a fixed substitution of letters, replacing them in some complex fashion
- After encrypting of a letter, the rotors advance positions, changing the substitution (to be applied to the next latter that is typed in)
- By this means, a rotor machine produces a complex polyalphabetic substitution cipher, which changes with every key press

Rotor Machines

- The example below shows schematically N = 3 rotors including some of their internal wiring
- Keyboard and dashboard are not shown
- The medium rotor advances its position after a full turn of the fast rotor, and the slow rotor advances its position after a full turn of the medium rotor
- Therefore, we have a N-stage polyalphabetic substitution algorithm
- For N = 5, there are 26^N (= 11881376) steps before a substitution is repeated!



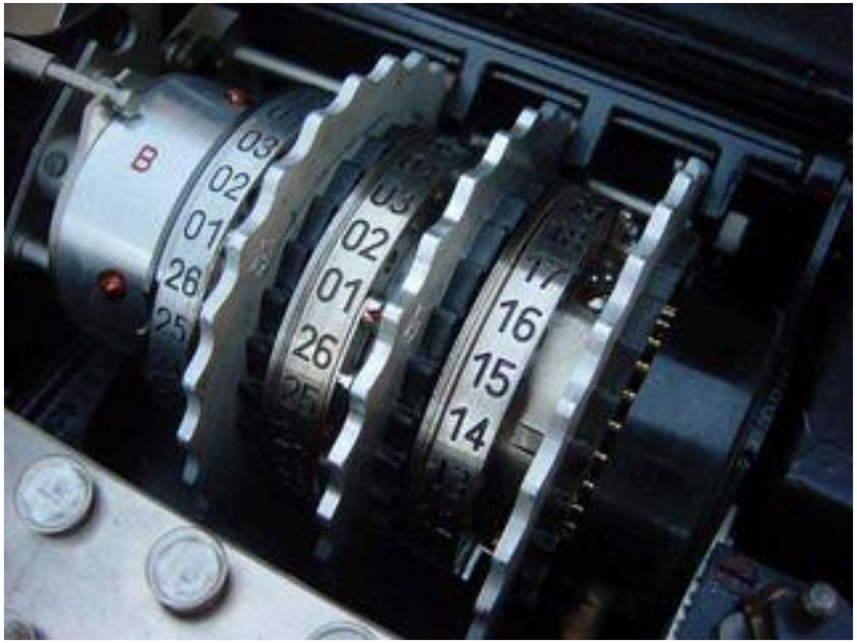
(a) Initial setting

A	24	21	26	20	1	8	A
B	25	3	1	1	2	18	B
C	26	15	2	6	3	26	C
D	1	1	3	4	4	17	D
E	2	19	4	15	5	20	E
F	3	10	5	3	6	22	F
G	4	14	6	14	7	10	G
H	5	26	7	12	8	3	H
I	6	20	8	23	9	13	I
J	7	8	9	5	10	11	J
K	8	16	10	16	11	4	K
L	9	7	11	2	12	23	L
M	10	22	12	22	13	5	M
N	11	4	13	19	14	24	N
O	12	11	14	11	15	9	O
P	13	5	15	18	16	12	P
Q	14	17	16	25	17	25	Q
R	15	9	17	24	18	16	R
S	16	12	18	13	19	19	S
T	17	23	19	7	20	6	T
U	18	18	20	10	21	15	U
V	19	2	21	8	22	21	V
W	20	25	22	21	23	2	W
X	21	6	23	9	24	7	X
Y	22	24	24	26	25	1	Y
Z	23	13	25	17	26	14	Z

(b) Setting after one keystroke

A	24	21	26	20	26	14	A
B	25	3	1	1	1	8	B
C	26	15	2	6	2	18	C
D	1	1	3	4	3	26	D
E	2	19	4	15	4	17	E
F	3	10	5	3	5	20	F
G	4	14	6	14	6	22	G
H	5	26	7	12	7	10	H
I	6	20	8	23	8	3	I
J	7	8	9	5	9	13	J
K	8	16	10	16	10	11	K
L	9	7	11	2	11	4	L
M	10	22	12	22	12	23	M
N	11	4	13	19	13	5	N
O	12	11	14	11	14	24	O
P	13	5	15	18	15	9	P
Q	14	17	16	25	16	12	Q
R	15	9	17	24	17	25	R
S	16	12	18	13	18	16	S
T	17	23	19	7	19	19	T
U	18	18	20	10	20	6	U
V	19	2	21	8	21	15	V
W	20	25	22	21	22	21	W
X	21	6	23	9	23	2	X
Y	22	24	24	26	24	7	Y
Z	23	13	25	17	25	1	Z

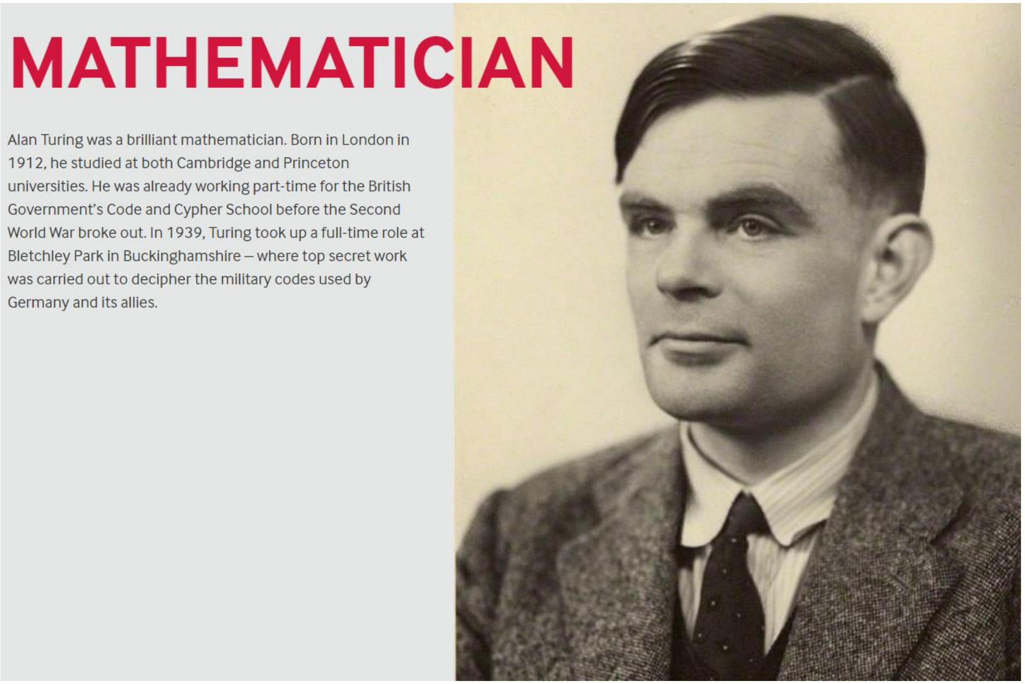
Example: The Enigma Machine



https://www.youtube.com/watch?v=-mdSvGUd0_c

How Alan Turing broke the Enigma Code

- https://www.iwm.org.uk/history/how-alan-turing-cracked-the-enigma-code
- The Imitation Game (Film, 2014)
- https://www.youtube.com/watch?v=nuPZUUED5uk



MATHEMATICIAN

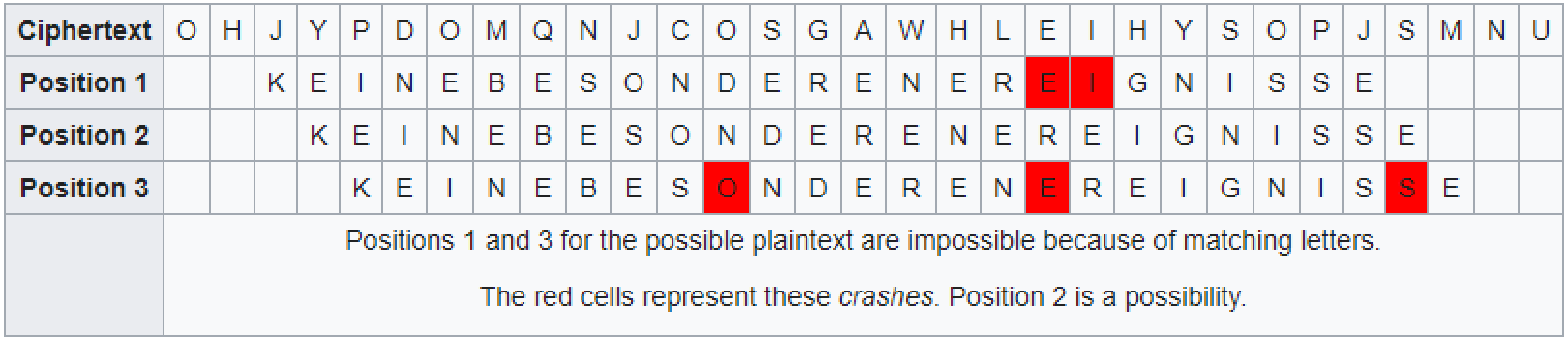
Alan Turing was a brilliant mathematician. Born in London in 1912, he studied at both Cambridge and Princeton universities. He was already working part-time for the British Government's Code and Cypher School before the Second World War broke out. In 1939, Turing took up a full-time role at Bletchley Park in Buckinghamshire – where top secret work was carried out to decipher the military codes used by Germany and its allies.

Breaking Enigma using Cribs

- Breaking Enigma was based on the following observations:
  - Plaintext messages were likely to contain certain phrases, e.g.
    - Weather reports contained the term ”WETTER VORHERSAGE”
    - Military units often sent messages containing “KEINE BESONDEREN EREIGNISSE”, i.e. “nothing to report”
  - A plaintext letter was never mapped onto the same ciphertext letter

Breaking Enigma using Cribs (Wikipedia)

- While the cryptanalysts did not know where exactly these cribs were placed in an intercepted message, they could exclude certain positions (i.e. Position 1 and 3):



Ciphertext	O	H	J	Y	P	D	O	M	Q	N	J	C	O	S	G	A	W	H	L	E	I	H	Y	S	O	P	J	S	M	N	U
Position 1			K	E	I	N	E	B	E	S	O	N	D	E	R	E	N	E	R	**E**	**I**	G	N	I	S	S	E				
Position 2				K	E	I	N	E	B	E	S	O	N	D	E	R	E	N	E	R	E	I	G	N	I	S	S	E			
Position 3					K	E	I	N	E	B	E	S	**O**	N	D	E	R	E	N	**E**	R	E	I	G	N	I	S	**S**	E		

Positions 1 and 3 for the possible plaintext are impossible because of matching letters.

The red cells represent these *crashes*. Position 2 is a possibility.

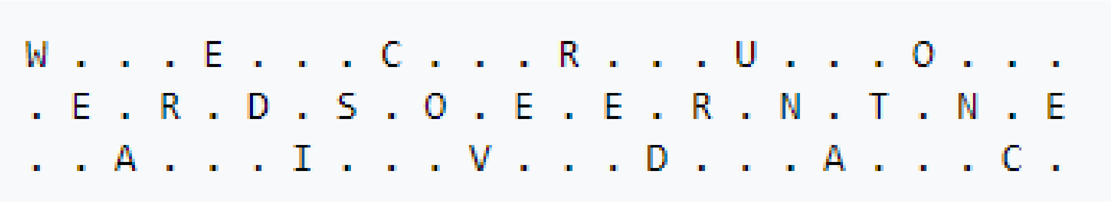
- From here on, possible rotor start positions and rotor wiring would be systematically examined using a “the bombe”, an electromechanical device designed by Alan Turing

Transposition Ciphers

- Now consider classical **transposition** or **permutation** ciphers
- These hide the message by rearranging the letter order without altering the actual letters used
- This can be recognised since ciphertext has the same frequency distribution as the original text

Rail Fence (Zigzag) Cipher

- Write message letters out diagonally up and down over a number of rows, then read off cipher row by row
- Example (Wikipedia): WE ARE DISCOVERED. RUN AT ONCE:



```
W . . . E . . . C . . . R . . . U . . . O . . .
. E . R . D . S . O . E . E . R . N . T . N . E
. . A . . . I . . . V . . . D . . . A . . . C .
```

- Resulting ciphertext:

```
WECRUOERDSOEERNTNEAIVDAC
```

Row Transposition Ciphers

- Write letters of message out in rows over a specified number of columns
- Then reorder the columns according to some key before reading off the columns
- Example:

```
Key:          4 3 1 2 5 6 7
Plaintext:    A T T A C K P
              O S T P O N E
              D U N T I L T
              W O A M X Y Z

Ciphertext:  TTNA APTM TSUO AODW COIX KNLY PETZ
```

Note that spaces are inserted to improve readability

Combined Ciphers

- Ciphers using substitutions or transpositions are not very strong because of language characteristics
- Hence consider using several ciphers in succession to make harder:
  - two substitutions make a more complex substitution
  - two transpositions make more complex transposition
  - but a substitution followed by a transposition makes a new much harder cipher
- Similar approaches are implemented in modern ciphers

Steganography

Steganography

- An alternative to encryption
- Steganography hides the existence of a message, by:
  - Using only a subset of letters / words in a longer message marked in some way
  - Using invisible ink
  - Hiding single bits at a time in suitable computer files (e.g., images or sound files)
- Drawback:
  - Not very economical in terms of overheads to hide a message (see also examples)

Example for Steganography

- Assume an x-by-y pixel wide image is stored in RGB format
- For each pixel the colour component (R, G and B) intensity is represented by a byte
- The image can be stored in a byte array of size [x][y][3]
- For each entry we change the least significant bit to hide bitwise a message, e.g.
-

R	G	B	becomes	R	G	B
01010110	11100101	10110000		01010111	11100100	10110000
11111111	10101001	00101010		11111111	10101000	00101011
11001101	10011001	11001010		11001100	10011001	11001010
…				…		

- This transformation allows the storage of the bit pattern 100101010, while causing minimal image distortions (invisible for the human eye)
- However, this method doesn't work in combination with image compression (e.g. JPEG compression)

https://stylesuxx.github.io/steganography/

Annex

1. Example cryptanalysis of a simple substitution cipher

Example Cryptanalysis of Simple Substitution Cipher

- Assume one intercepts the ciphertext below
- We know (out of the context) that
  - the plaintext message is written in English
  - The message has been encoded using the simple substitution cipher
- Intercepted ciphertext:
  UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXA
  IZVUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSXEPYE
  POPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ
- We do a frequency analysis of the ciphertext and begin with the most common letters in the English language, e and t
- Guess ciphertext letters P & Z are plaintext letters e and t (we use small letters to distinguish between both):
  UtQSOVUOHXMOeVGeOteEVSGtWStOeFeESXUDBMETSXAItV
  UEeHtHMDtSHtOWSFeAeeDTSVeQUZWYMXUtUHSXEeYEeOeD
  tStUFeOMBtWeFUetHMDJUDTMOHMQ

Example Cryptanalysis

- □ Guess (!) Z?P means *the*:

  UtQSOVUOHXMOeVGeOteEVSGtWStOeFeESXUDBMET
  SXAItVUEeHtHMDtSHtOWSFeAeeDTSVeQUZWYMXUtUH
  SXEeYEeOeDtStUFeOMBt**W**eFUetHMDJUDTMOHMQ

- □ Assume W is *h*:

  UtQSOVUOHXMOeVGeOteEVSGt**h**StOeFeESXUDBMETS
  XAItVUEeHtHMDtSHtO**h**SFeAeeDTSVeQUZWYMXUtUHSX
  EeYEeOeDtStUFeOMBt**h**eFUetHMDJUDTMOHMQ

Example Cryptanalysis

- □ Guess word *that*, translating S into a:
  UtQSOVUOHXMOeVGeOteEVSG***thSt***OeFeESXUDBMET
  SXAItVUEeHtHMDtSHtOhSFeAeeDTSVeQUZWYMXUtUH
  SXEeYEeOeDtStUFeOMBtheFUetHMDJUDTMOHMQ
- □ Ciphertext becomes:
  UtQ**a**OVUOHXMOeVGeOteEV**a**G***that***OeFeE**a**XUDBMET
  aXAItVUEeHtHMDt**a**HtOhsFeAeeDT**a**VeQUZWYMXUtUH
  aXEeYEeOeDt**a**tUFeOMBtheFUetHMDJUDTMOHMQ

Example Cryptanalysis

- □ Guess that AeeD means *been*:
  UtQaOVUOHXMOeVGeOteEVaGthatOeFeEaXUDBMETaXAItVUEeHtHMDtaHtOhsFe**AeeD**TaVeQUZWYMXUtUHaXEeYEeOeDtatUFeOMBtheFUetHMDJUDTMOHMQ
- □ Resulting in (with A→b and D→n):
  UtQaOVUOHXMOeVGeOteEVaGthatOeFeEaXUnBMETaX**b**ItVUEeHtHM**n**taHtOhsFe**been**TaVeQUZWYMXUtUHaXEeYEeOe**n**tatUFeOMBtheFUetHM**n**JU**n**TMOHMQ

Example Cryptanalysis

- Is HMntaHt meaning *contact*?

  UtQaOVUOHXMOeVGeOteEVaGthatOeFeEaXUnBMET aXbltVUEeHt**HMntaHt**OhsFebeenTaVeQUZWYMXUtUH aXEeYEeOentatUFeOMBtheFUetHMnJUnTMOHMQ

- Therefore (with H→ c and M→ o):

  UtQaOVUO**c**X**o**OeVGeOteEVaGthatOeFeEaXUnBoETa XbltVUEe**ctcontact**OhaFebeenTaVeQUZWY**o**XUtU**c**aXEe YEeOentatUFeO**o**BtheFUet**co**nJUnT**o**O**co**Q

Example Cryptanalysis

- Does VUEect mean *direct*?
  UtQaOVUOcXoOeVGeOteEVaGthatOeFeEaXUnBoETaX
  bIt**VUEect**contactOhaFebeenTaVeQUZWYoXUtUcaXEeY
  EeOentatUFeOoBtheFUetconJUnToOcoQ
- Therefore (with V→ d, U → i and E→ r):
  itQaO**di**OcXoOe**d**GeOterdaGthatOeFeraXinBorTaXbIt
  directcontactOhaFebeenTadeQiZWYoXiticaXreYreOent
  atiFeOoBtheFietconJinToOcoQ

Example Cryptanalysis

- □ Does GeOterdaG mean yesterday?
  itQaOdiOcXoOed**GeOterdaG**thatOeFeraXinBorTaXbIt
  directcontactOhaFebeenTadeQiZWYoXiticaXreYreOent
  atiFeOoBtheFietconJinToOcoQ
- □ Therefore (with G→ y and O → s):
  itQasdiscXosed**yesterday**thatseFeraXinBorTaXbItdirect
  contactshaFebeenTadeQiZWYoXiticaXreYresentatiFeso
  BtheFietconJinToscoQ

Example Cryptanalysis

- Moscow calling? itQasdiscXosedyesterdaythatseFeraXinBorTaXbltdirectcontactshaFebeenTadeQiZWYoXiticaXreYresentatiFesoBtheFietconJin**ToscoQ**
- Therefore (with T → m and Q → w): it**w**asdiscXosedyesterdaythatseFeraXinBor**m**aXbltdirectcontactshaFebeen**m**ade**w**iZWYoXiticaXreYresentatiFesoBtheFietconJin**moscow**

Example Cryptanalysis

- X means *l*, F means *v*, B means *f*?
  itwas**discXosed**yesterdaythat**seFeraXinBormaX**bltdirectcontactshaFebeenmadewiZWYoXiticaXreYresentatiFesoBtheFietconJinmoscow
- Therefore:
  itwas**disclosed**yesterdaythat**severalinformal**bltdirectcontactshavebeenmadewiZWYoliticalreYresentativesofthevietconJinmoscow

Example Cryptanalysis

- I means *u*, Z means *t*, W means *h*, Y means *p*?
  itwasdisclosedyesterdaythatseveralinformal**bIt**directcontactshavebeenmade**wiZW**YoliticalreYresentativesofthevietconJinmoscow

- Therefore:
  itwasdisclosedyesterdaythatseveralinformal**but**directcontactshavebeenmade**with**politicalrepresentativesofthevietconJinmoscow

Example Cryptanalysis

- Finally: J means *g*:
  itwasdisclosedyesterdaythatseveralinformalbutdirectcontactshavebeenmadewithpoliticalrepresentativesofthe**vietconJ**inmoscow
- Therefore (with spaces added):
  it was disclosed yesterday that several informal but direct contacts have been made with political representatives of the vietcong in moscow

CT437
COMPUTER SECURITY AND FORENSIC COMPUTING

BLOCK CIPHERS

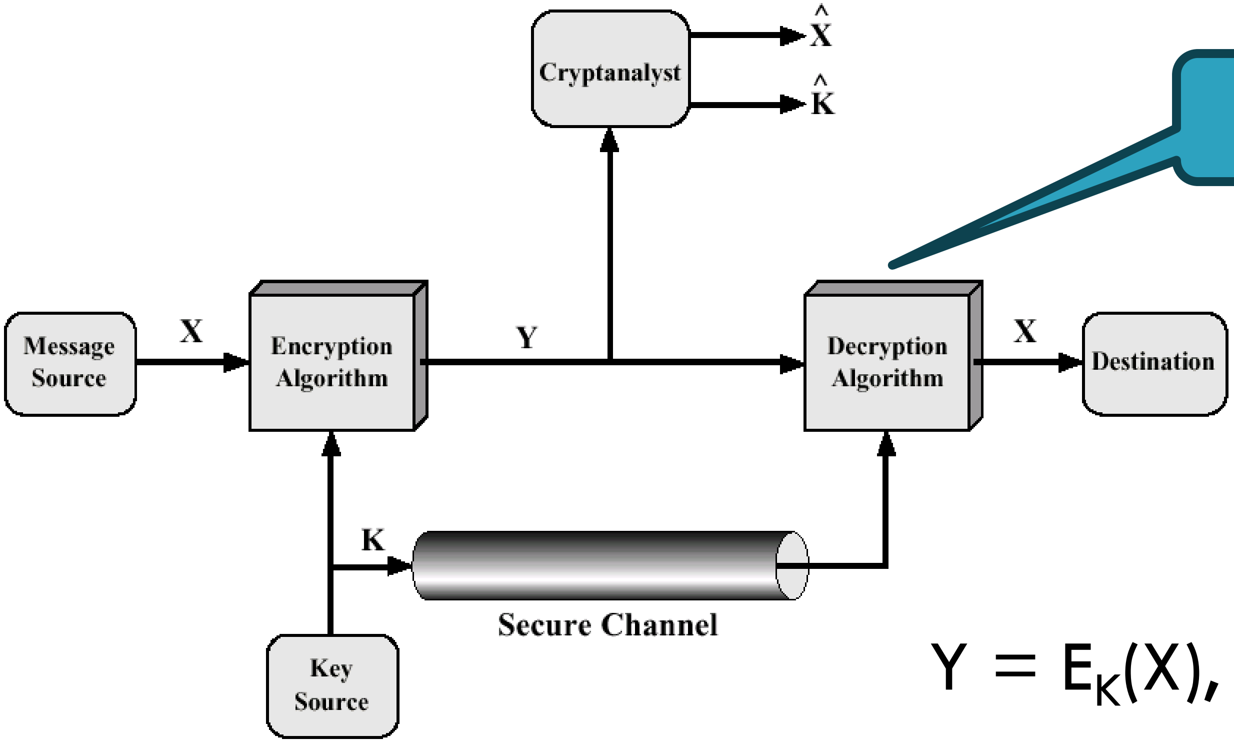
Dr. Michael Schukat

Ollscoil na Gaillimhe
University of Galway

Lecture Overview

- This lecture provides an introduction to one of the fundamental building blocks to provide confidentiality, namely **block ciphers**, thereby covering the following:
  - Symmetric versus Public Key Algorithms
  - Block ciphers versus Stream Ciphers
  - Building Blocks of modern Block Ciphers
  - Modes of operation of block ciphers
  - Examples for modern block ciphers

Recall: Model of Conventional Cryptosystem



Symmetric Algorithm!

$$Y = E_K(X),\ X = E_K^{-1}(Y)$$

Symmetric Key Algorithms

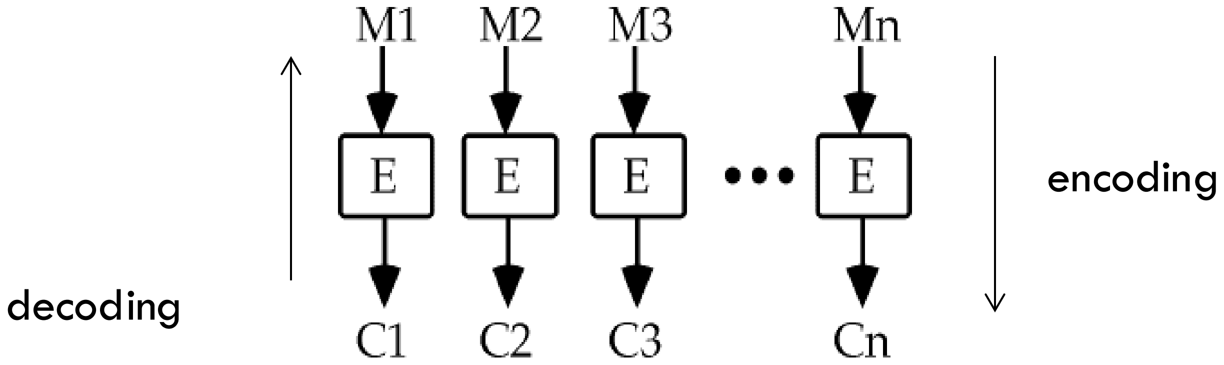
- Also called ciphers for traditional / conventional / single key / private key encryption
- Here the encryption key can be calculated from the decryption key and vice versa
  - Normally both keys are the same
- The algorithm / cipher itself is public, i.e. is not a secret
- If the key is disclosed, communications are compromised
- The key is also **symmetric**, parties are equal
- Hence methods does not protect sender from receiver forging a message & claiming is sent by sender → nonrepudiation is usually not provided

Public-Key Algorithms

- Also called ciphers for two key / asymmetric cryptography
- These involve the use of two keys:
  - a **public key**, which may be known by anybody, and can be used to encrypt messages, and verify signatures (later!)
  - a **private key**, known only to the recipient/owner, used to decrypt messages, and sign (create) signatures
- The keys are **asymmetric**, because they are not equal
- While the public and its private key are interlinked, it is mathematically very hard to recover the private key via its public key
- Public key algorithms are generally significantly slower that symmetric algorithms, therefore these are often used to securely convey symmetric algorithm' (session) keys
- More later!

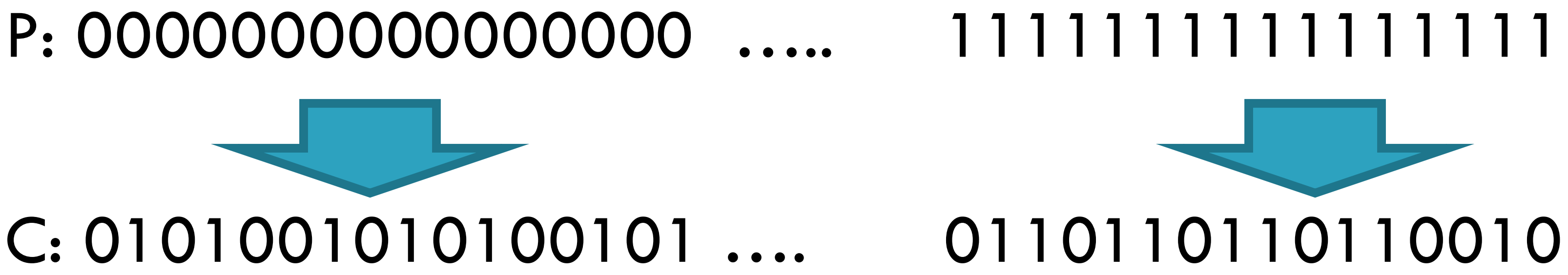
Block Ciphers versus Stream Ciphers

- In a block cipher the data (e.g. text, video, or a network packet) to be encrypted is broken into blocks M1, M2, etc. of K bits length, each of which is then encrypted
- The encryption process is like a substitution on very big characters – 64 bits or more



- In contrast, **stream ciphers** (→ next lecture) only process one bit or one byte at a time

Example Block Cipher Transformation



P: 0000000000000000 ….. 1111111111111111

C: 0101001010100101 …. 0110110110110010

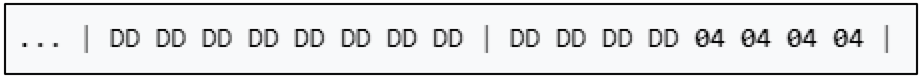
- □ Block size K is 16 bits
- □ If there wasn't a cipher available for this transformation, we'd require a table with 2^{16} entries
  - → Not feasible
- □ Note that there are $(2^{16})!$ possible substitutions

Block Ciphers and Padding

- Messages are usually not multiples of K bits
- Padding is a way to take data that may or may not be a multiple of the block size for a cipher and extend it out so that it is
  - It is only applied to the last block that is being encrypted
- Padding must be reversable, i.e., one must be able to distinguish between relevant content and padding bytes in a block

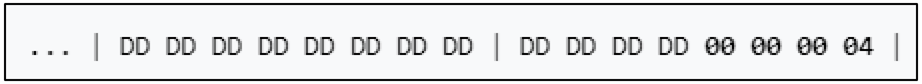
Padding Algorithms

- Let N be the number of bytes required to make a final block of data the same size as the block size
  - PKCS7 padding works by appending N bytes with the binary value of N; example:



```
... | DD DD DD DD DD DD DD DD | DD DD DD DD 04 04 04 04 |
```

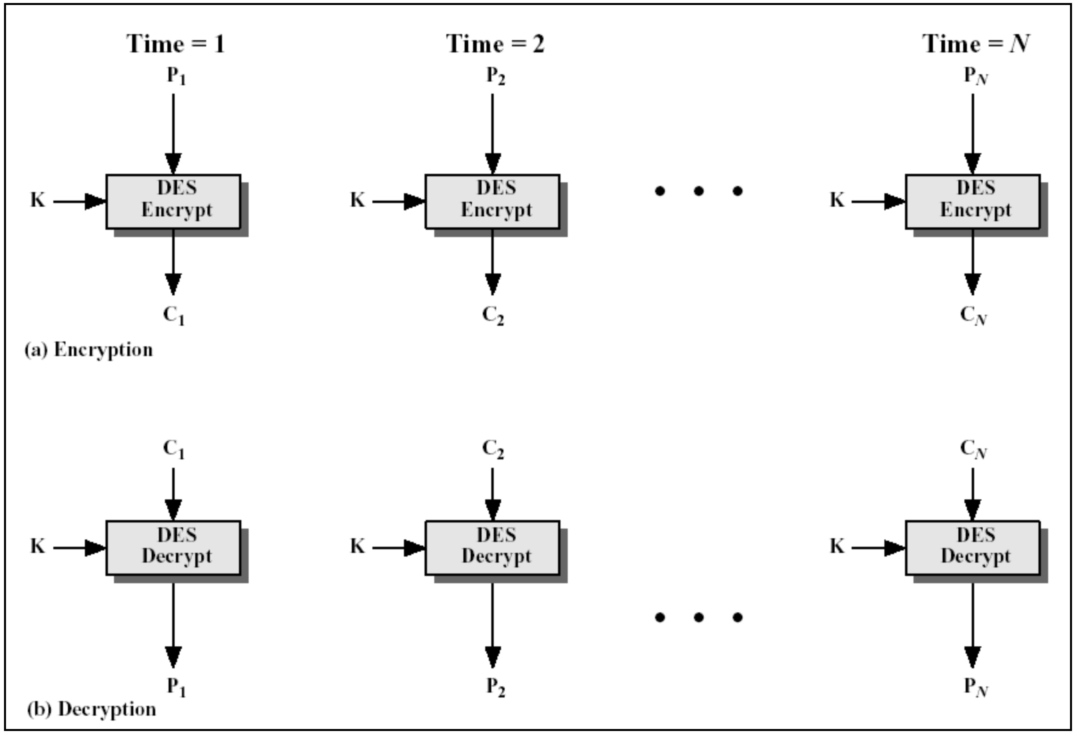
  - ANSI X9.23 padding works by appending N-1 bytes with the value of 0 and a last byte with the value of the binary value of N; example:



```
... | DD DD DD DD DD DD DD DD | DD DD DD DD 00 00 00 04 |
```

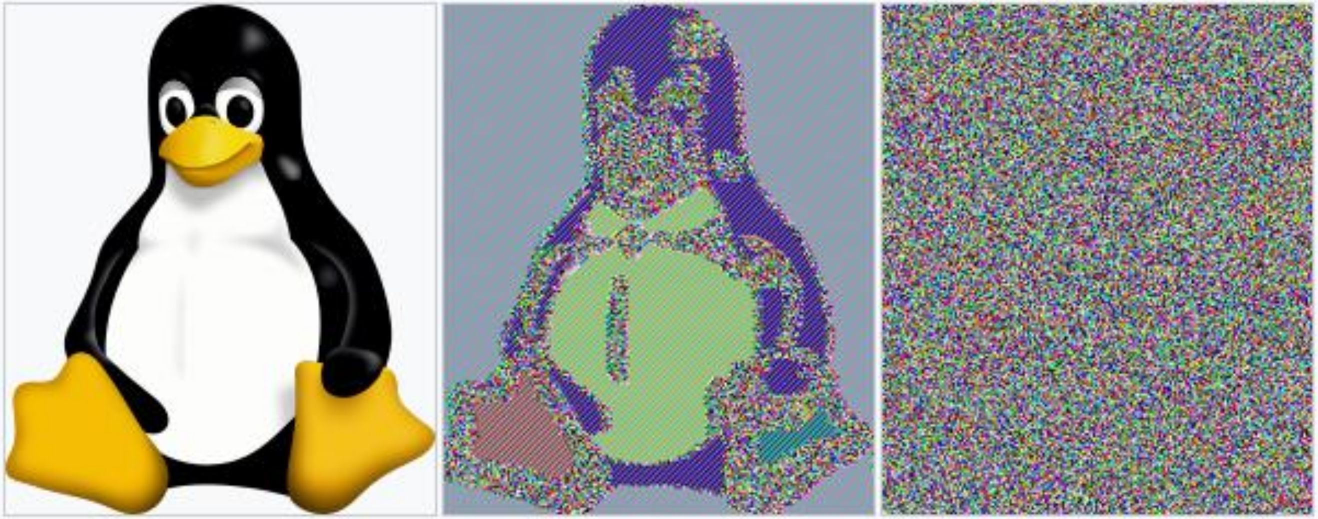
Modes of Operation: Electronic Codebook (EBC) Mode

- These modes comprise different strategies on how to use block ciphers (to encode messages)
- What are the advantages / disadvantages of the ECB mode?
- Note that "DES" in the diagram on the right is just an example for a block cipher



Characteristics and Limitations of ECB Mode

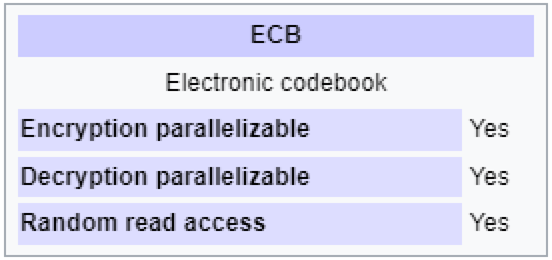
Source: Wikipedia



Original image

Using ECB allows patterns to be easily discerned

Modes other than ECB result in pseudo-randomness

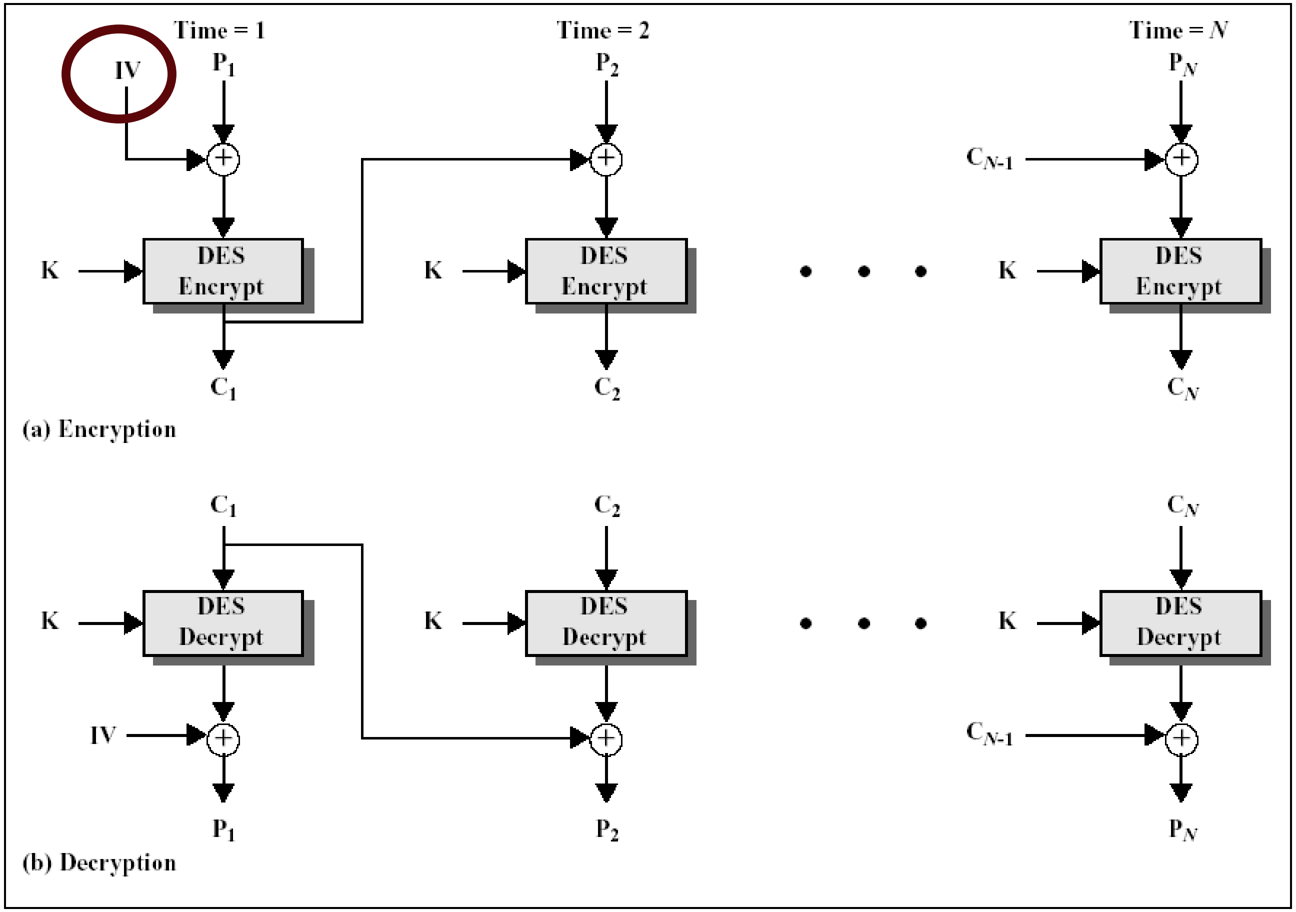


ECB	
Electronic codebook	
Encryption parallelizable	Yes
Decryption parallelizable	Yes
Random read access	Yes

Why would one avoid the Electronic Codebook Mode?

- In ECB mode identical plaintext blocks result in identical ciphertext blocks
- An attacker, while not able to decode the ciphertext blocks, would conclude that the encoded data is repetitive / structured, i.e. could be an image
- However, random data (e.g. long cryptographic keys) could be still encoded in ECB
    - E.g., a 512-bit key could be encrypted using a 128-bit block cipher using ECB mode

Modes of Operation: Cipher Block Chaining (CBC) Mode

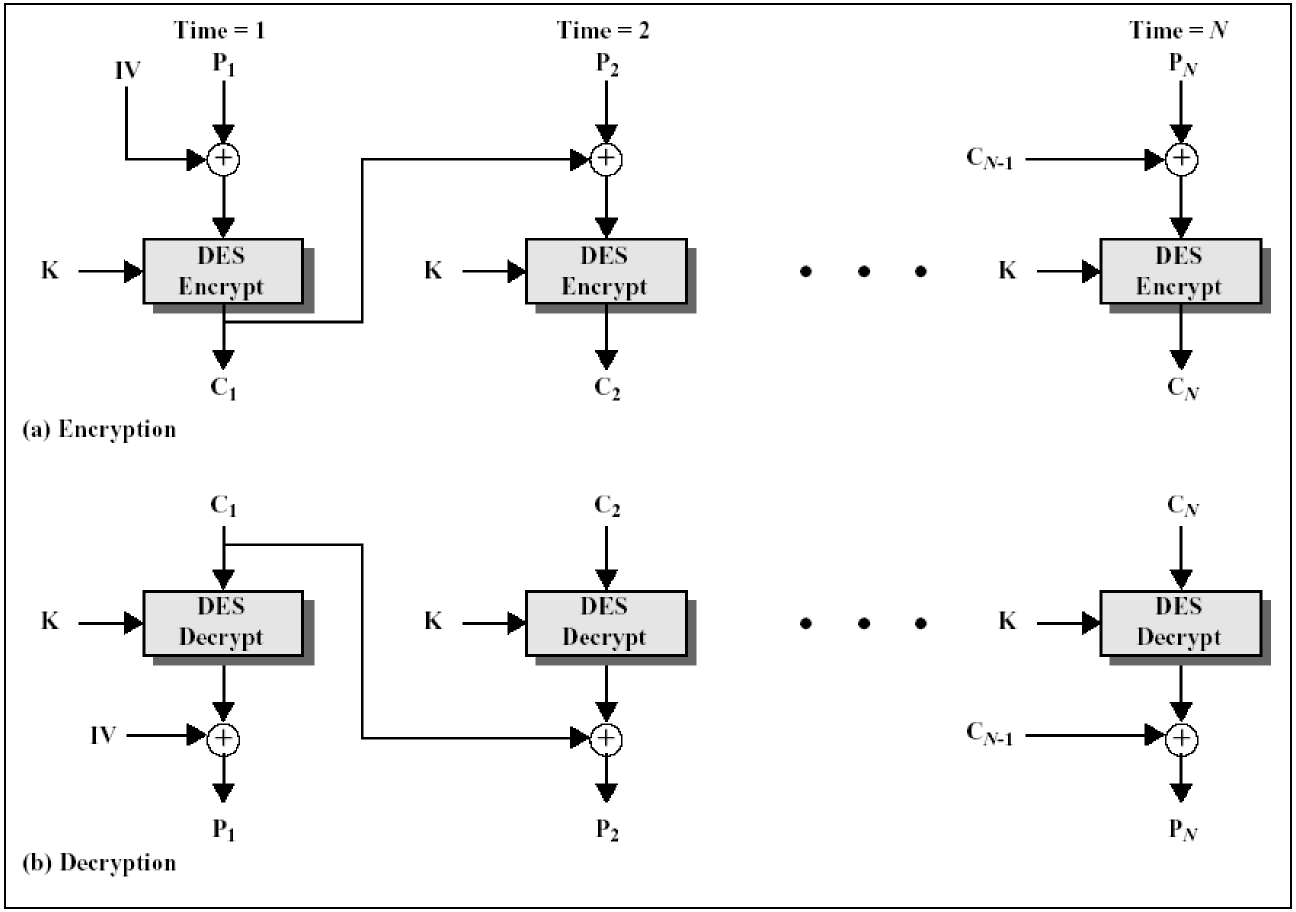


The Initialisation Vector (IV)

- An IV is a block of bits that is used by several modes (including CBC) to randomise the encryption
- An initialization vector has different security requirements than a key, so the IV usually does not need to be secret
- For most block cipher modes it is important that an initialisation vector is never reused under the same key, i.e. it must be a **cryptographic nonce**
  - Hence distinct ciphertexts are generated even if the same plaintext is encrypted multiple times using the same key
- In data communication, the IV may be attached as plaintext to the encrypted data, and send to the receiver

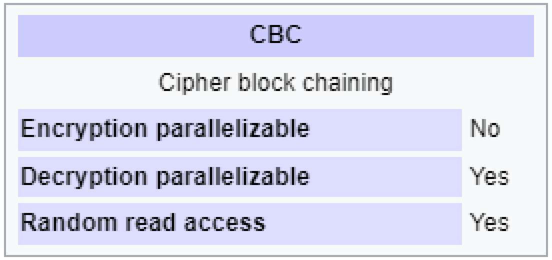
Modes of Operation: Cipher Block Chaining (CBC) Mode

- What are the characteristics of the CBC mode?

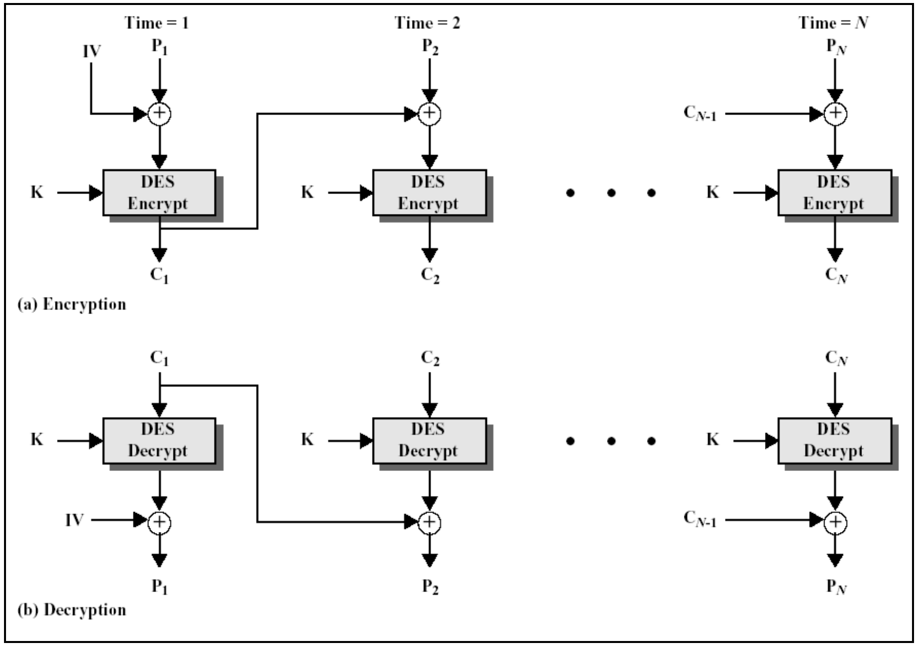


Modes of Operation: Cipher Block Chaining (CBC) Mode

- For encryption, a one-bit change in a plaintext or IV affects all following ciphertext blocks
- Decrypting with the incorrect IV causes the first block of plaintext to be corrupt but subsequent plaintext blocks will be correct
  - This could be problematic if the IV was kept a secret and is used to expand the algorithm's key space

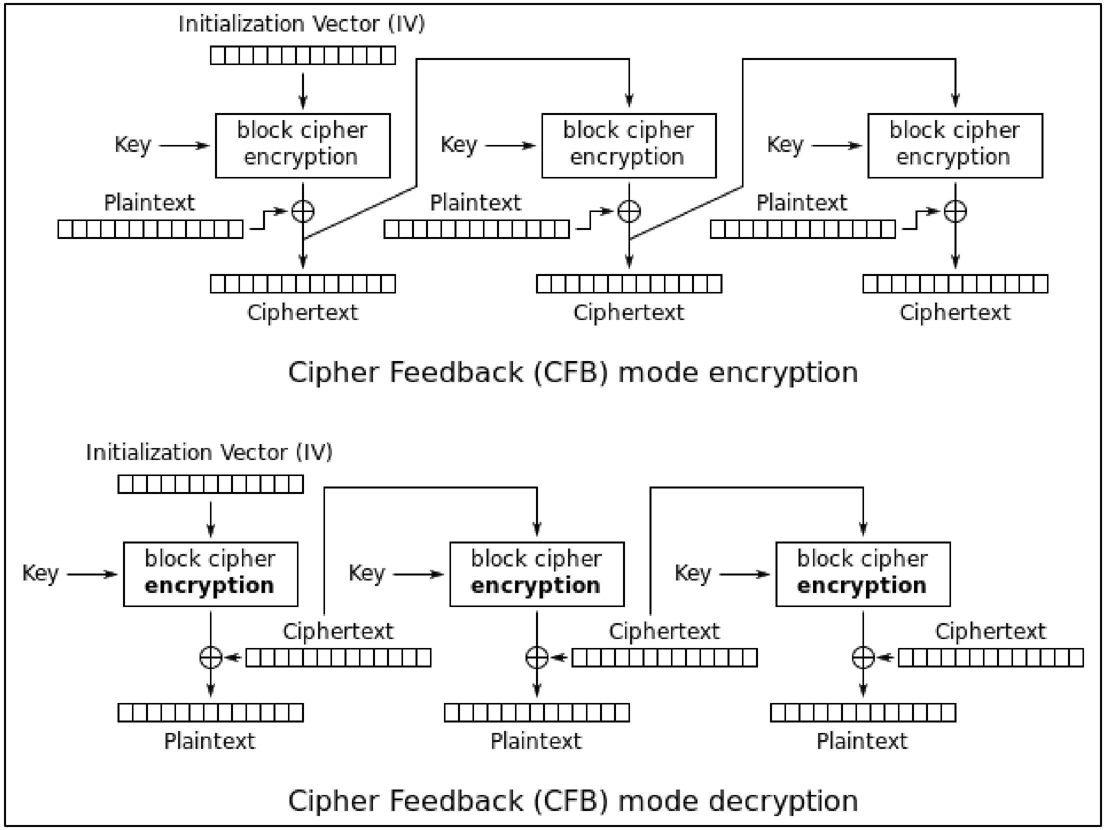


CBC	
Cipher block chaining	
Encryption parallelizable	No
Decryption parallelizable	Yes
Random read access	Yes



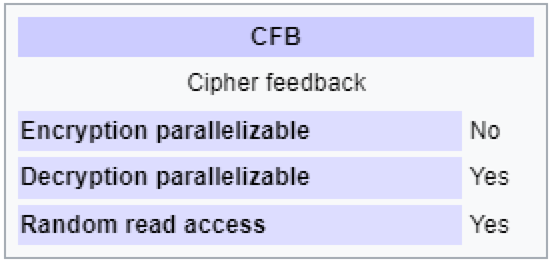
Full Block Cipher Feedback (CFB) Mode

- Note that CFB only requires block encryption for both encoding and decoding

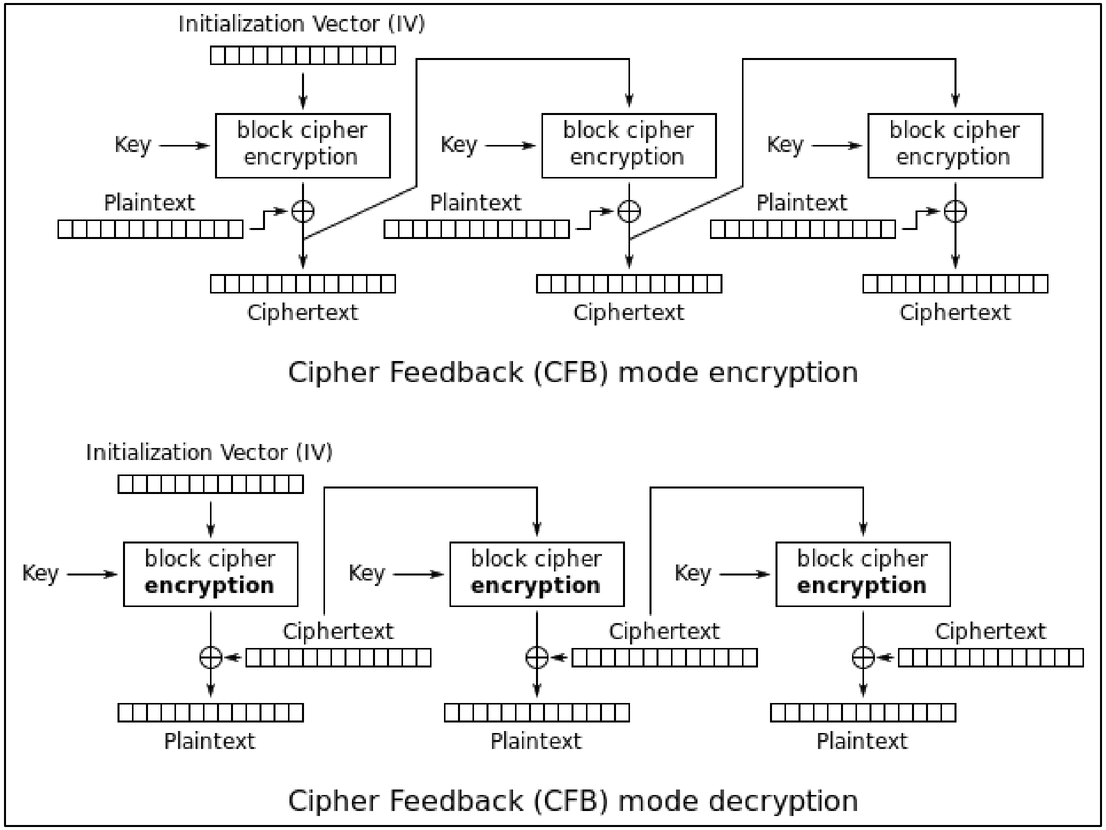


Full Block Cipher Feedback (CFB) Mode

- This mode is particularly useful, when decryption needs to be fast (i.e., parallelisable)
- Note that CFB only requires block encryption for both encoding and decoding



CFB	
Cipher feedback	
Encryption parallelizable	No
Decryption parallelizable	Yes
Random read access	Yes

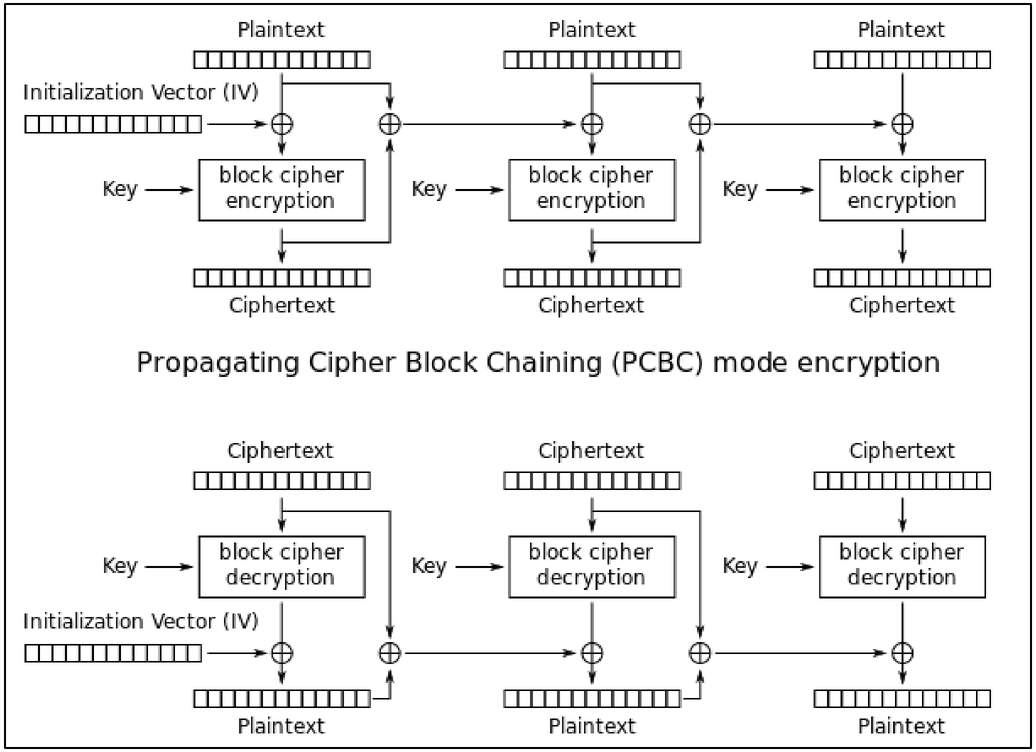


Cipher Feedback (CFB) mode encryption

Cipher Feedback (CFB) mode decryption

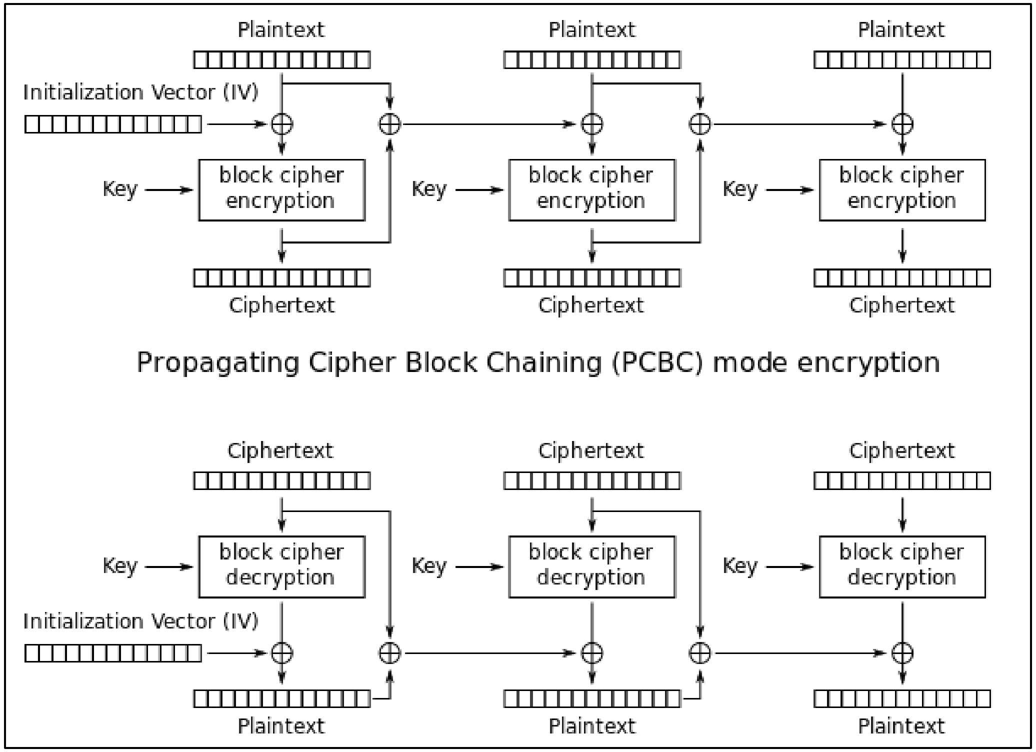
Propagating Cipher Block Chaining (PCBC) Mode

- This mode fixes the IV problem of CBC, i.e., decrypting PCBC with the incorrect IV causes all blocks of plaintext to be corrupt

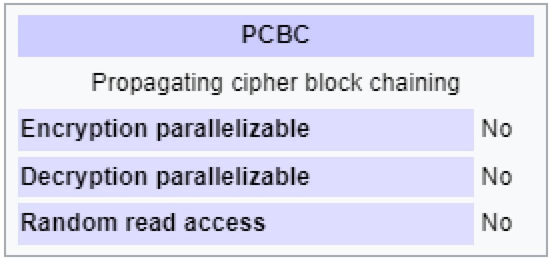


Propagating Cipher Block Chaining (PCBC) Mode

- This mode fixes the IV problem of CBC, i.e., decrypting PCBC with the incorrect IV causes all blocks of plaintext to be corrupt



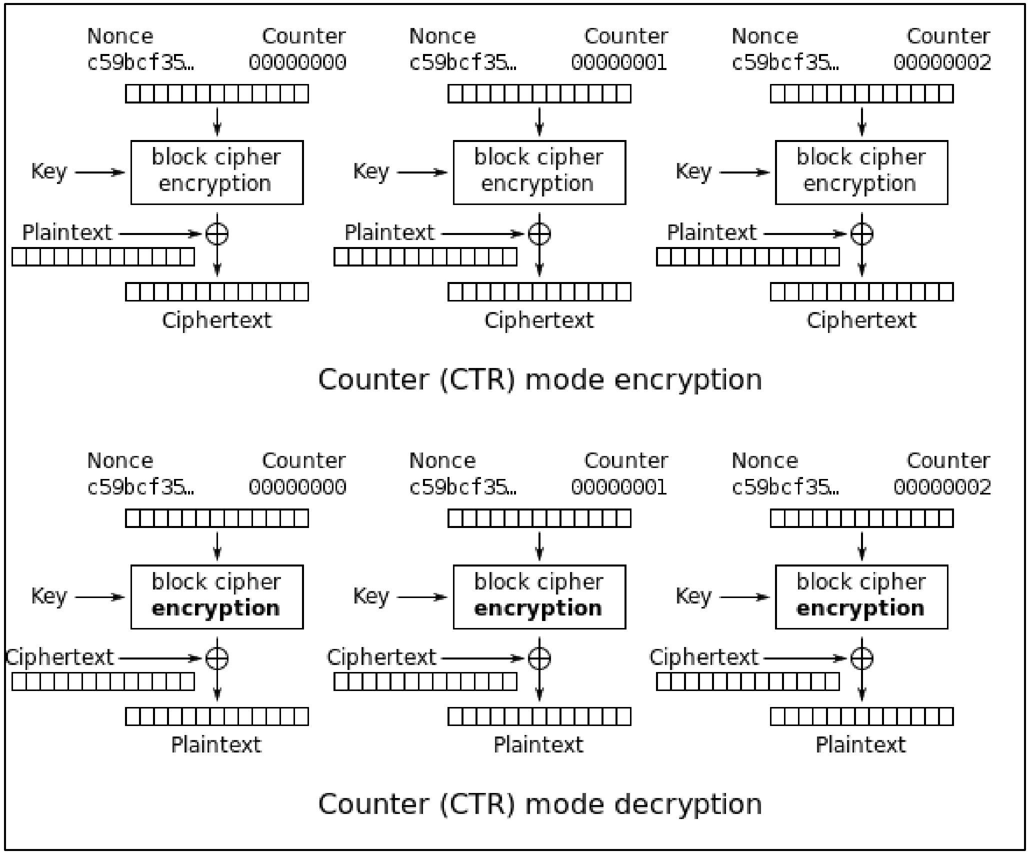
Propagating Cipher Block Chaining (PCBC) mode encryption



PCBC	
Propagating cipher block chaining	
Encryption parallelizable	No
Decryption parallelizable	No
Random read access	No

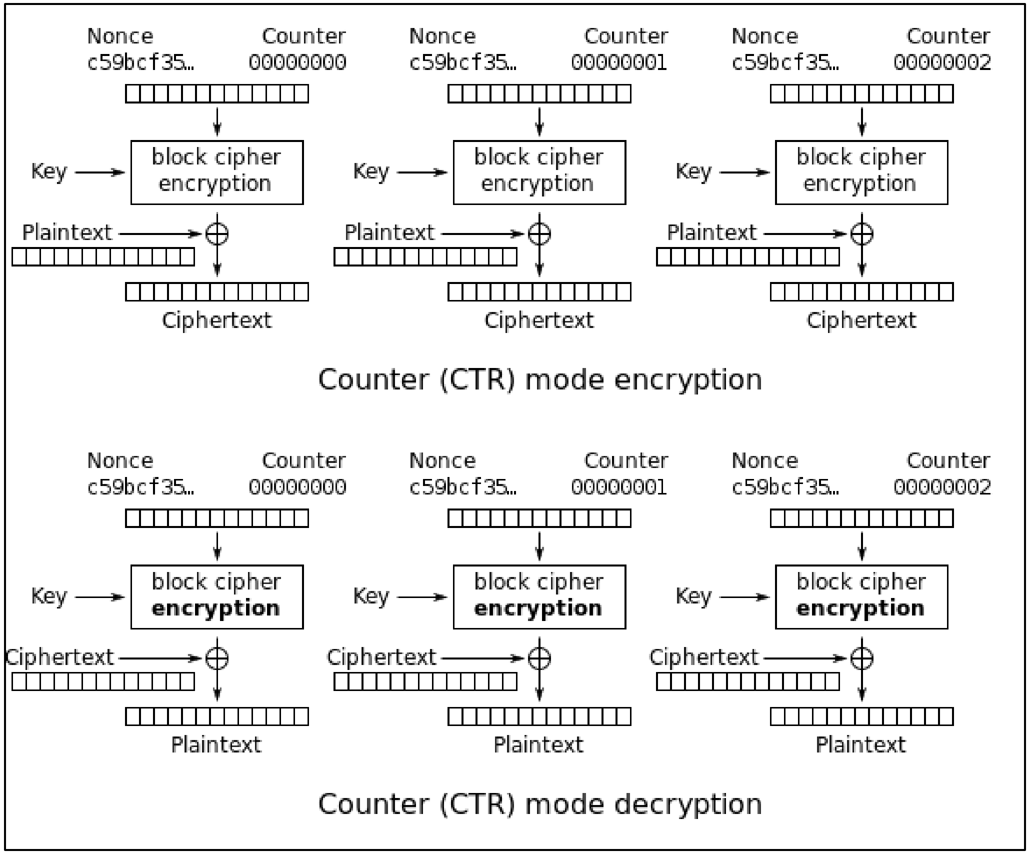
Counter (CTR) Mode

- Here the random nonce (which is equivalent to an IV) is complemented with an incremented counter value
- Note that CFB only requires block encryption for both encoding and decoding



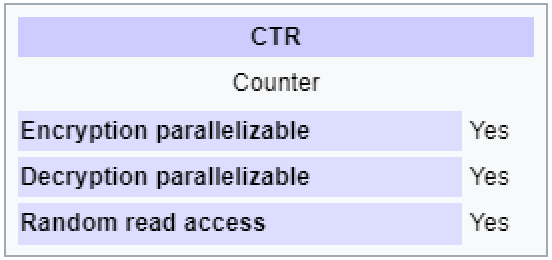
Counter (CTR) Mode

- Here the random nonce (which is equivalent to an IV) is complemented with an incremented counter value
- Note that CFB only requires block encryption for both encoding and decoding



Counter (CTR) mode encryption

Counter (CTR) mode decryption



CTR	
Counter	
Encryption parallelizable	Yes
Decryption parallelizable	Yes
Random read access	Yes

Another Question …

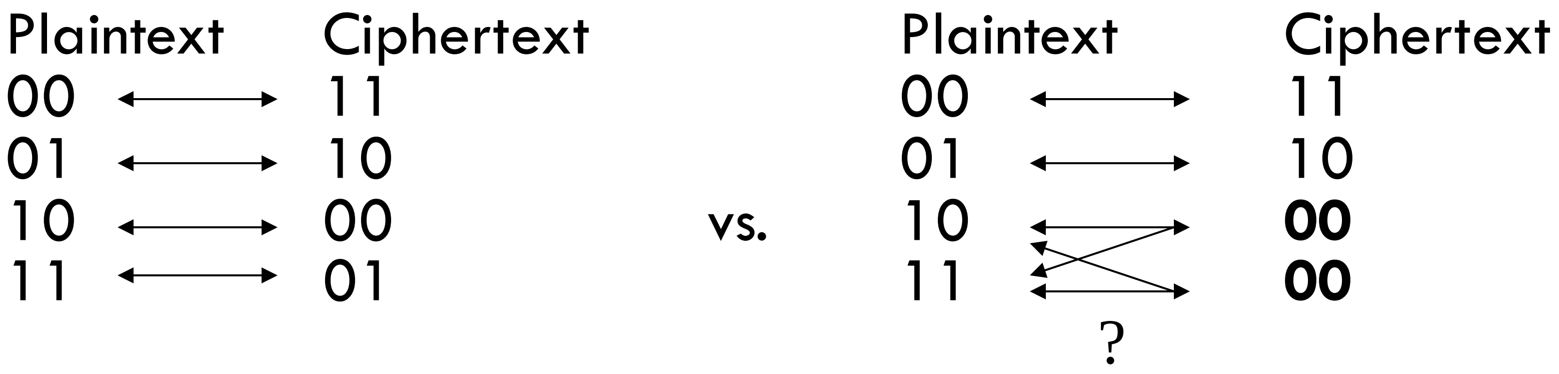
- Assume you have a large data file stored on your computer that needs to be encrypted / decrypted on-the-fly, potentially with random block access
- You pick a suitable block cipher for this task
- **Which mode of operation would you choose?**

Building Block Ciphers

- Confusion, diffusion and the avalanche effect
- Using SP-Networks
- Using Feistel Networks

Important Block Cipher Principle

- Transformations must be reversible ("non-singular"), e.g.,



Plaintext		Ciphertext
00	↔	11
01	↔	10
10	↔	00
11	↔	01

vs.

Plaintext		Ciphertext
00	↔	11
01	↔	10
10	↔	**00**
11	↔	**00**

?

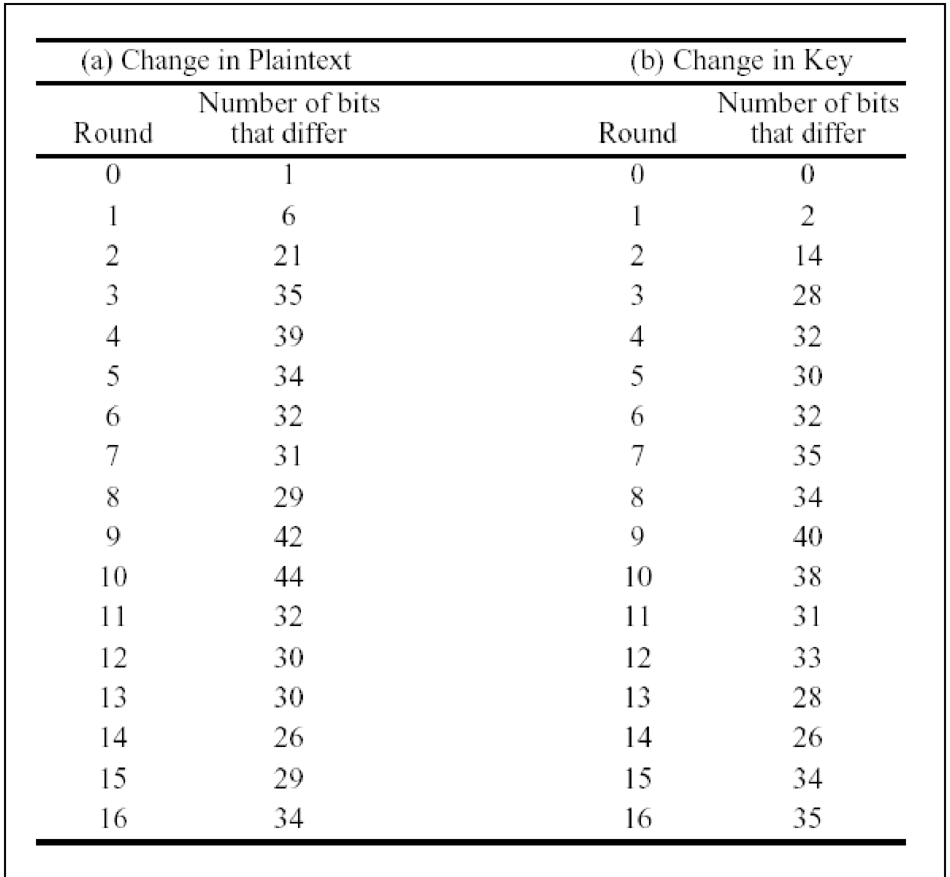
- There must be a 1:1 association between a n-bit plaintext and an-bit ciphertext, otherwise mapping (encryption) is irreversible

Confusion and Diffusion

- A block cipher needs to completely obscure the statistical properties of original message (obviously)
- Claude Shannon introduced two terms:
  - **Diffusion** seeks to make the statistical relationship between the plaintext and ciphertext as complex as possible
  - **Confusion** seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible
- Both thwart attempts to deduce the key
- Thos can be achieved, by applying a cryptographic operation iteratively multiple times, i.e. over multiple rounds
  - See also the avalanche effect (next slide)

The Avalanche Effect

- Practically, confusion and diffusion provide for encryption algorithms, where a slight change in either the key or the plaintext results in a significant change in the ciphertext
- The table shows some characteristics of the DES algorithm (later), that encrypts a block over 16 rounds
- A swap of a single bit either in the key or in the plaintext results in an incrementally growing change in the ciphertext (avalanche effect)



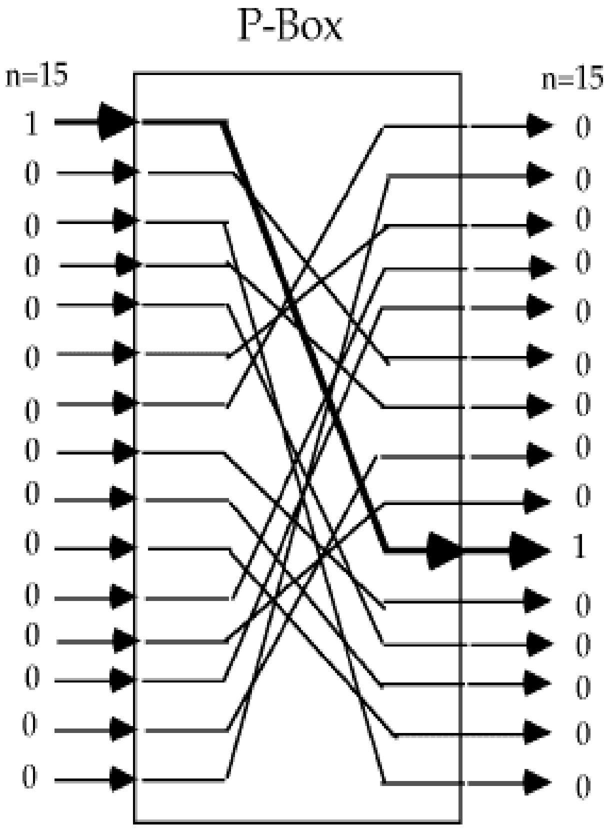
(a) Change in Plaintext		(b) Change in Key	
Round	Number of bits that differ	Round	Number of bits that differ
0	1	0	0
1	6	1	2
2	21	2	14
3	35	3	28
4	39	4	32
5	34	5	30
6	32	6	32
7	31	7	35
8	29	8	34
9	42	9	40
10	44	10	38
11	32	11	31
12	30	12	33
13	30	13	28
14	26	14	26
15	29	15	34
16	34	16	35

Block Cipher Building Blocks

- In order to build a block cipher efficiently, one needs robust building blocks, that can be easily implemented and tested
  - The robustness of the block cipher depends on the robustness of its components
- These blocks are combined or iterated through creating a block cipher
- The most common building blocks are:
  - P-Boxes
  - S-Boxes
  - Feistel ciphers

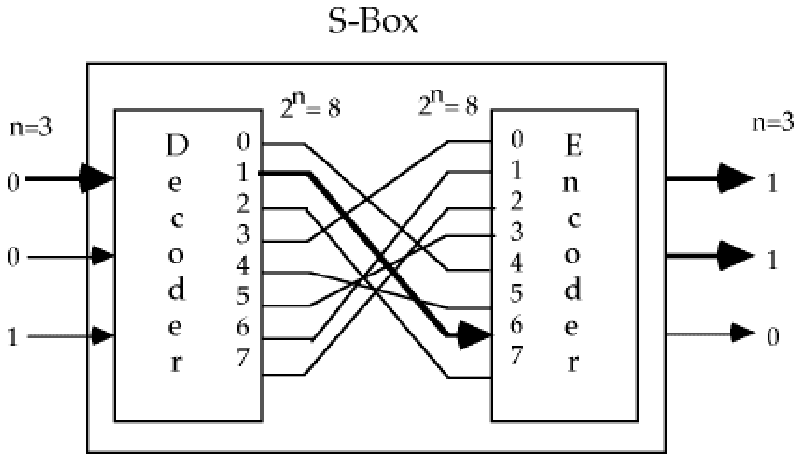
The Permutation Operation

- A binary word (i.e. block) has its bits reordered (permuted)
  - Similar to classical transposition ciphers
- This operation is represented by a **P-box** (see diagram)
- Here the re-ordering / internal wiring forms the key
- The example shown allows for 15! = 1,307,674,368,000 combinations

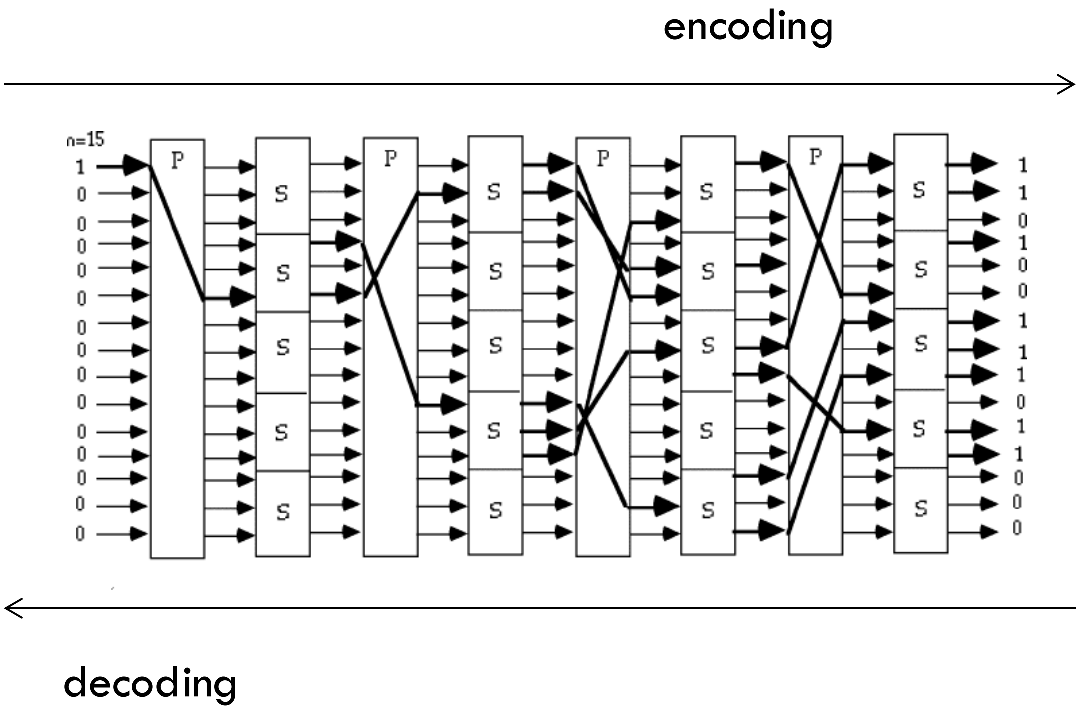


The Substitution Operation

- A binary word is replaced by some other binary word
- Similar to classical substitution ciphers
- This operation is represented by an **S-box**
- Here the re-ordering / internal wiring forms the key
- The box shown allows
  for 8! = 40320 combinations



Substitution-Permutation Network



- The key describes the internal wiring of all S-boxes and P-boxes
- The same key can be used for encoding and decoding, hence it is a **private key encryption algorithm**
- The direction of the process determines encoding / decoding

Question:

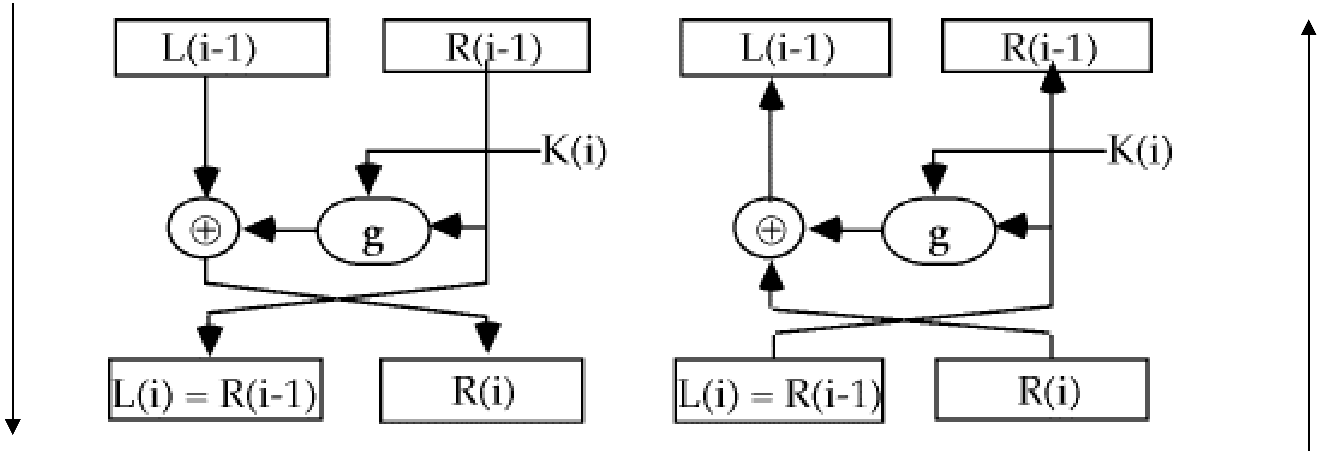
- How big is the key space for this arrangement?
- How many bits are needed to describe a single S or P box?
- What is the total number of bits required to describe all boxes?

The Feistel Cipher

- In practice, we need algorithms that can decrypt and encrypt messages using similar code / hardware for both
- An S-P network as seen in the example cannot be easily reversed when implemented in hardware / software
  - i.e. one needs different functions for encoding / decoding
- In contrast, a **Feistel cipher** is an invertible cipher structure which adapts Shannon's S-P network in an easily invertible structure for encoding and decoding
  - In fact, it can use other cryptographic building blocks
- It is based on the concept of the **invertible product cipher**
- It was invented by Horst Feistel, who worked at IBM Thomas J Watson Research Labs in early 70's

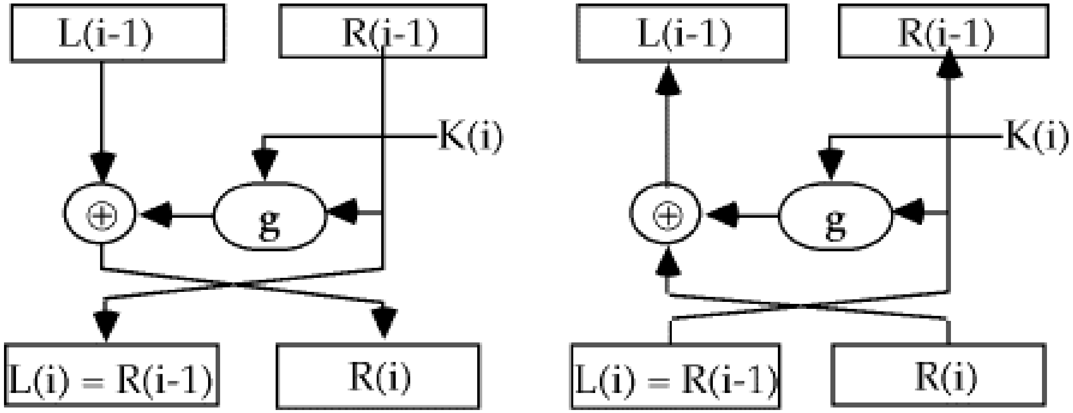
The Feistel Cipher – A Single Round

- The idea is to partition the input block into two halves, L(i−1) and R(i−1), and use only R(i−1) in the ith round (part) of the cipher
- The function g incorporates the equivalent of one stage of the S-P network, controlled by part of the key K(i) known as the ith subkey



The Feistel Cipher – A single Round

- A round of a Feistel cipher can be described functionally as:
  - L(i) = R(i-1)
  - R(i) = L(i-1) EXOR g(K(i), R(i-1))



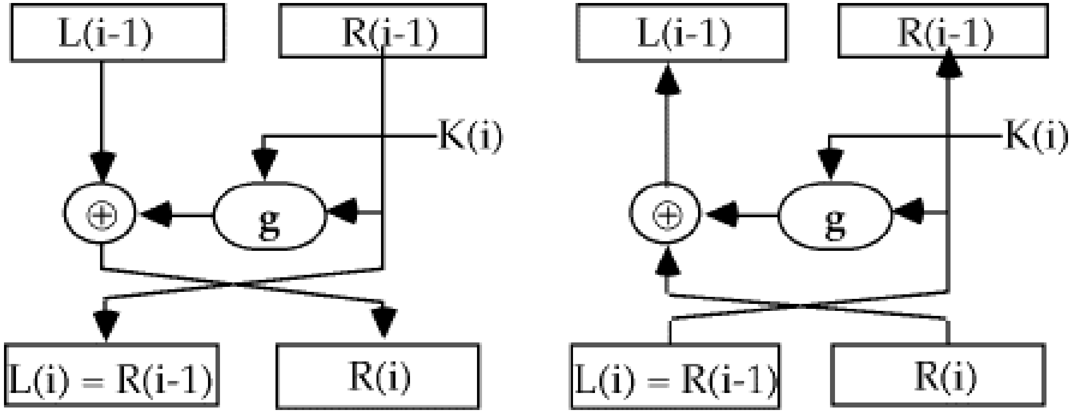
Recap: Symmetry of Bitwise EXOR

- A EXOR B = C
  A EXOR C = B
  C EXOR B = A

	0	1
0	0	1
1	1	0

In-Class Activity: Feistel Cipher – Single Round

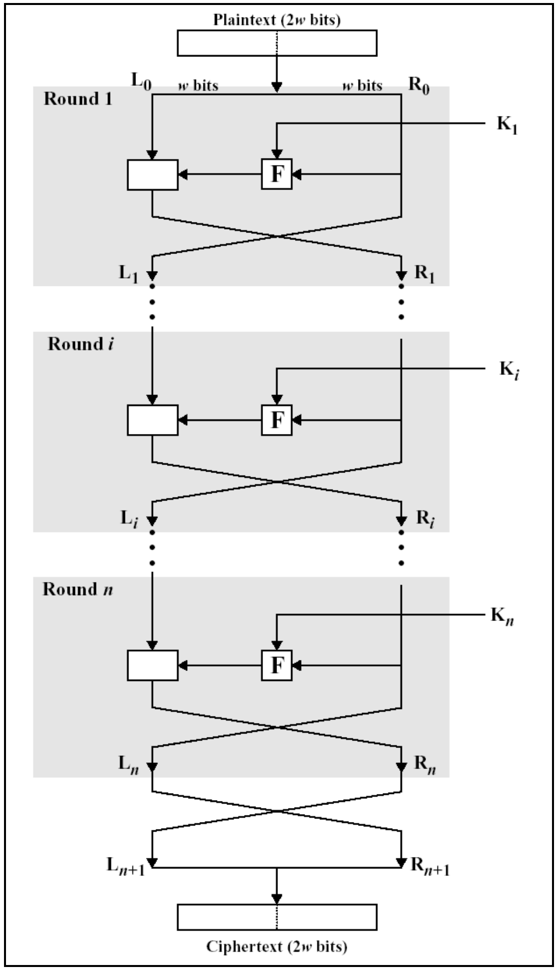
- Encoding of 01011110:
  - L(i – 1) = 0101 R(i – 1) = 1110
  - g(K(i), R(i-1)) = 1001 **L(i) = ?**
  - **R(i) = ?**
  - **Therefore 01011110 becomes ?**



Example Feistel Cipher – Single Round

- Encoding of 01011110:
  - L(i − 1) = 0101 R(i − 1) = 1110
  - g(K(i), R(i−1)) = 1001 L(i) = 1110
  - R(i) = 0101 XOR 1001 = 1100
  - Therefore 01011110 becomes 11101100
- Decoding of 11101100:
  - L(i) = 1110 R(i) = 1100
  - g(K(i), R(i−1)) = 1001 R(i − 1) = 1110
  - L(i − 1) = 1100 XOR 1001 = 0101
  - Therefore 1110 1100 becomes 01011110

Feistel Network



- Common structure of many modern block ciphers
- It perform multiple transformations (single rounds) sequentially, whereby output of i^{th} round becomes the input of the $(i+1)^{th}$ round
- Every round gets is own subkey, which is derived from master key
- Decryption process goes from bottom to top

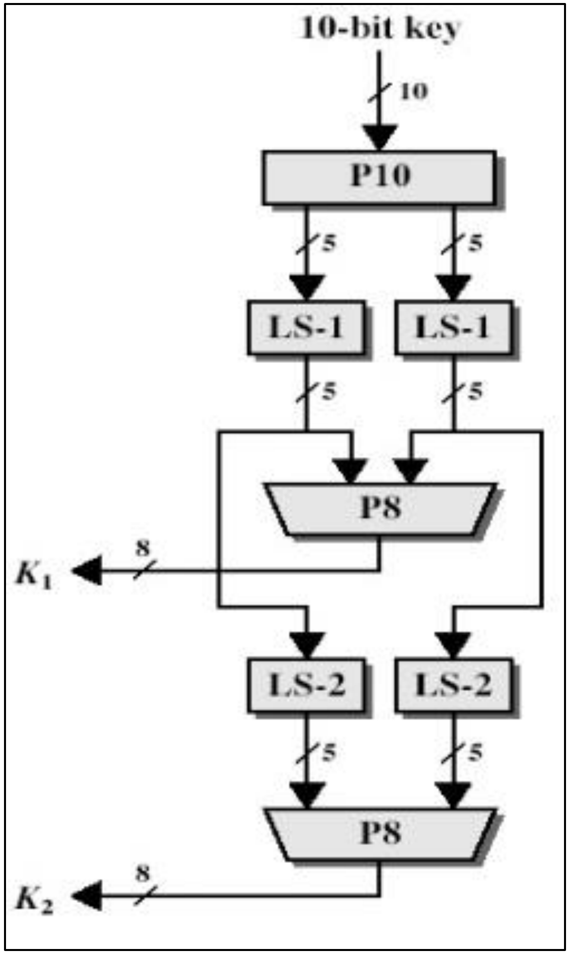
Feistel Cipher Design Elements

These include

- Block size (typically 64 – 256 bits)
- Key size (typically 80 – 256 bits)
- Number of rounds (typically > 16)
- Subkey generation algorithm
- Round function

Simple Methods for Subkey Generation

- Here two 8-bit round keys (K_1 and K_2) are derived from a 10-bit (master) key:
  - The 10-bit master enters the permutation box (P10)
  - The output is split into 2 parts
  - Each part is left-rotated by one bit (LS-1)
  - Both parts are concatenated and passed into a permutation box (P8)
  - P8 has eight outputs, which make K_1



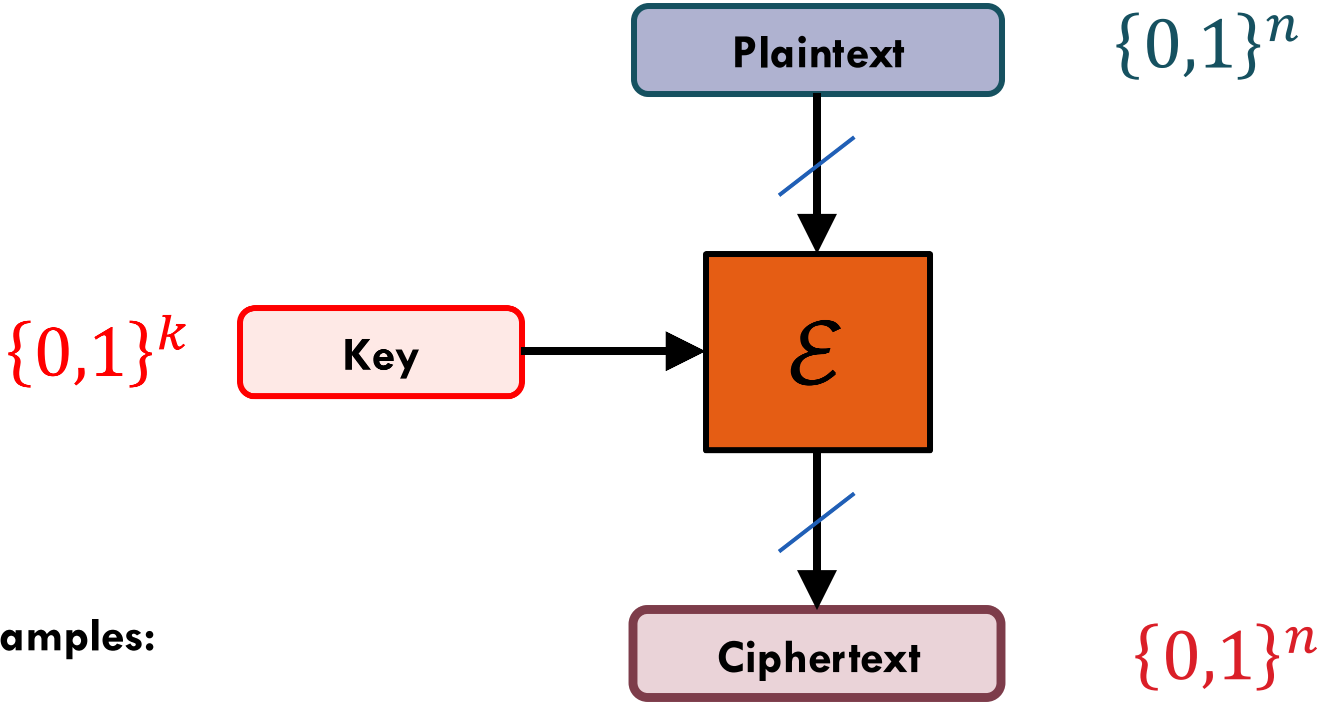
Block Cipher Examples

- Data Encryption Standard (DES)
- AES

Common Block Cipher Key and Block Lengths

$Key\ length\ k = 80, 128, 192, 256$

$Block\ length\ n = 64, 128, 256$



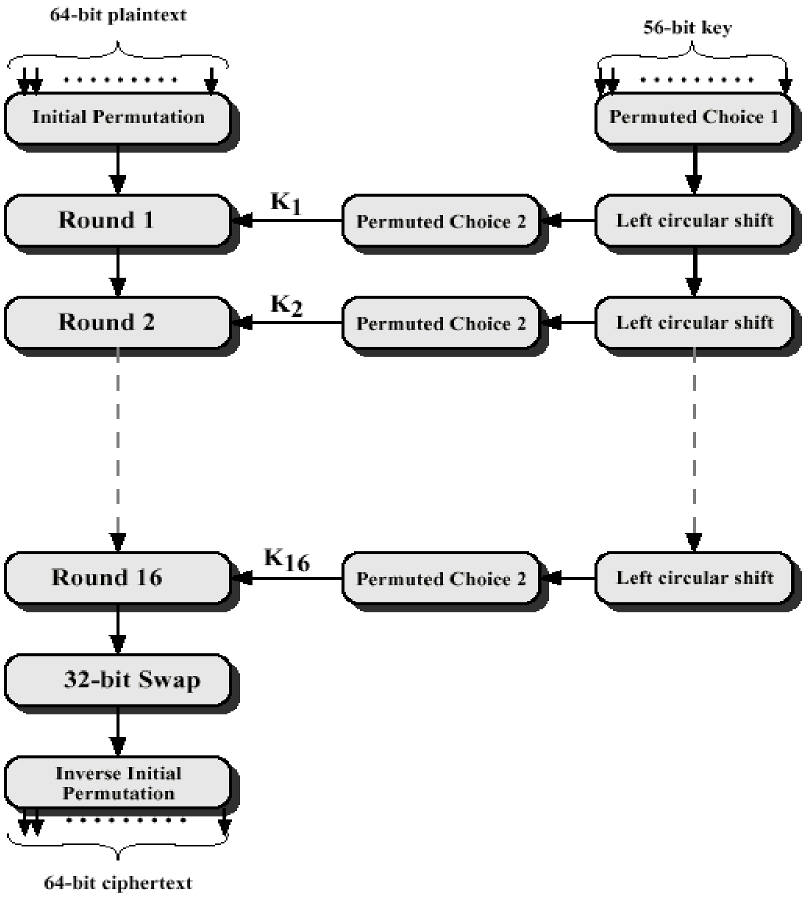
Examples:

DES: $k = 56,\ \ n = 64$

AES: $k = 128, 192, 256,\ \ n = 128$

Data Encryption Standard (DES)

- □ DES was the first block cipher widely used in industry
- □ Introduced in 1976
- □ 64-bit block length
- □ 56-bit key length
- □ Feistel network with 16 rounds and 48-bit subkeys



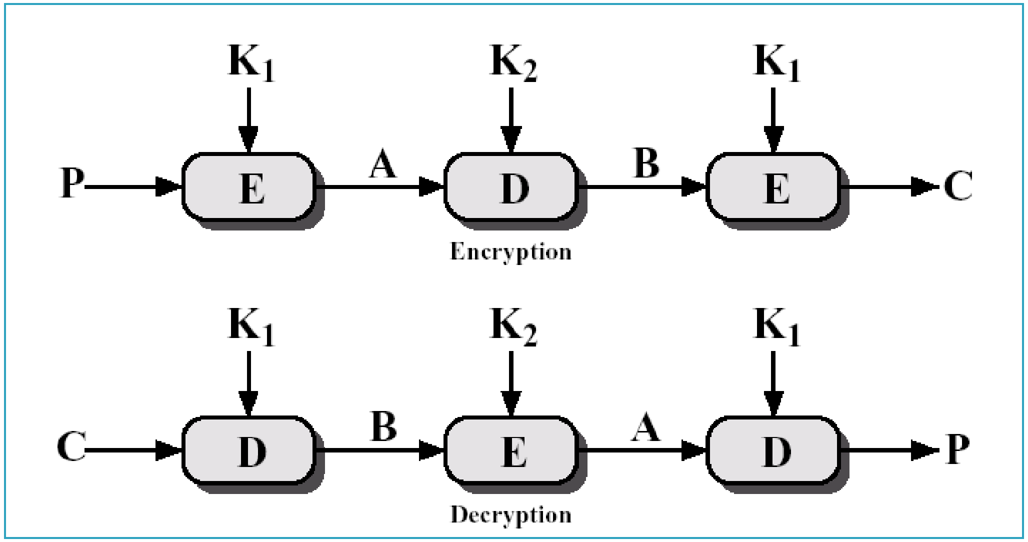
The DES Challenge

- □ Contest to demonstrate to the US government that 56-bit DES is an ineffective form of encryption
- □ The goal of the challenge was to decrypt secret messages which had been encrypted with DES

Name	When	Duration	Hardware used
DES I Challenge	June 1997	140 days	Up to 70,000 PC
DES II Challenge	February 1998	41 days	?
DES Challenge II-2	July 1998	56 hours	Custom FPGA Design
DES Challenge III	January 1999	22 hours	~100,000 PC

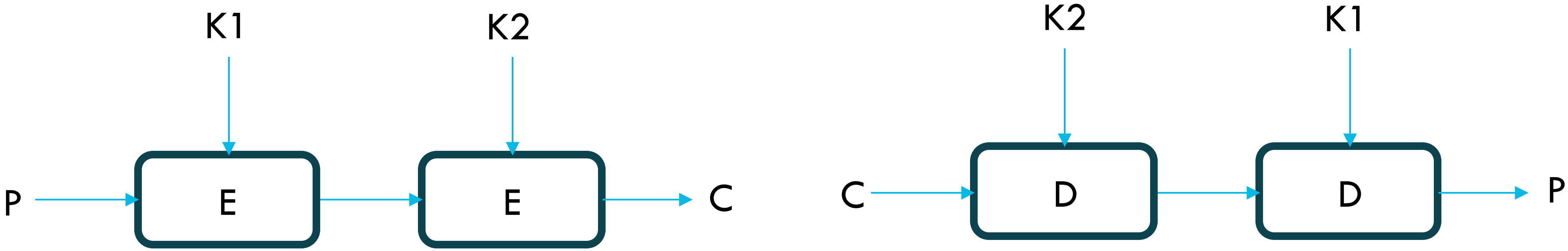
Triple DES

- DES has been widely used and implemented across many OS and crypto libraries, so attempts were made to increase its active life span
- This resulted in Triple-DES
- It is based on three processing stages
- Note the symmetry in the encoding and decoding process
- In principal, this concatenation can be applied to every private key block cipher
- There are 2 common keying options:
  - 2 keys (as shown in the figure)
  - 3 keys (one for each stage)



Double-DES and the Meet-in-the-Middle Attack

- Double DES uses two instances of DES with different keys



- While this algorithm uses two independent keys, it is not as sound as it looks
- It is vulnerable to the meet-in-the-middle attack, where an attacker has access to P and C, and tries to determine K1 and K2

Double-DES and the Meet-in-the-Middle Attack

- This attack is an example of a space-time tradeoff, where the adversary does the following:
  1. Encrypt P using every possible key, and copy each key and the resulting cyphertext into a table T1
     - T1 will have 2 columns and 2^{56} rows
  2. Decrypt C using every possible key, and copy each key and the resulting plaintext into a table T2
     - Again, T2 will have 2 columns and 2^{56} rows
  3. Check for identical ciphertext / plaintext entries in T1 and T2
  4. Their corresponding keys K1 and K2 are key candidates and can be further validated using other plaintext/cyphertext pairs
- Overall, this process requires 2^{56} encryption and 2^{56} decryption attempts, so overall 2 x 2^{56} = 2^{57} attempts (rather than 2^{112} attempts) are required
- Note that this attack can also be applied to Triple DES, but it would require 2^{2x56} attempts

Advanced Encryption Standard (AES)

- Successor of DES since 2002
- Based on a S-P network
- Block size is 128-bit
- Key length is configurable can be 128, 192 or 256 bit
- Stronger & faster than Triple-DES
  - 2 * 56! << 128!
- Envisaged active life until ~2030
- Full specification & design details public
- Algorithm has reference implementations across many programming languages

Breaking Block Ciphers

Why does Block and Key Length matter?

- Cryptographic algorithms with short block length can be tackled as seen with the substitution cipher
- Large keys and large blocks prevent **brute-force attacks / searches**
  - Take the ciphertext and try all possible key combinations (or block permutations), until the text is successfully decoded (e.g. until the decryption provides meaningful text)

Brute Force Search / Attacks

- DES uses 56-bit key has a key space that contains 2^{56} (= 7.2×10^{16}) keys
  - Deemed unsafe since the 1990s
- Triple-DES uses two 56-bit keys. and its key space contains 2^{112} (= 5.1×10^{33}) keys
  - Its use will be prohibited from 2024!
- AES-128 key space contains 2^{112} (= 3.4×10^{38}) keys
  - Generally accepted minimum key length today
- Top secret information requires the use of either AES-192 or AES-256

Brute Force Search / Attacks

- □ Always possible to simply try every key
- □ Most basic attack, effort proportional to key size
- □ Assume that you either know or recognise plaintext
- □ GPUs are very good at this task, for example a single RTX 3070 GPU can crack a DES key in ~215 days

Key Size (bits)	**Number of Alternative Keys**	**Time required at 1 decryption/µs**	**Time required at 10^6 decryptions/µs**
32	$2^{32} = 4.3 \times 10^9$	2^{31} µs = 35.8 minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	2^{55} µs = 1142 years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	2^{127} µs = 5.4×10^{24} years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	2^{167} µs = 5.9×10^{36} years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	2×10^{26} µs = 6.4×10^{12} years	6.4×10^6 years

Side-Channel Attacks

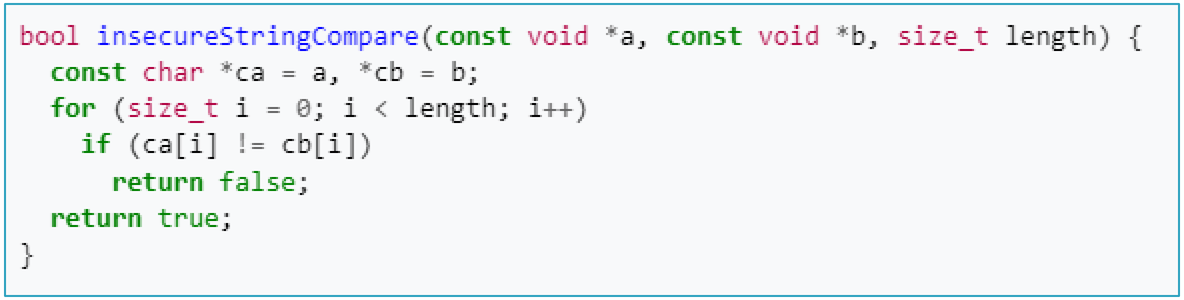
- AES is cryptographically sound and there is no practical cryptographic "break" that is faster than a brute-force attack
- However, there are possible side-channel attacks
- Generally, these are attacks on implementations of a cipher on hardware or software systems that inadvertently leak data, e.g.
  - Timing information (how long does an encryption take)
  - Cache and memory content (→ HeartBleed)

Timing Attacks

- □ Here the attacker attempts to compromise a cryptosystem by analysing the time taken to execute a cryptographic algorithm
- □ Every logical operation in a computer takes time to execute, and the time can differ based on the input
- □ With precise measurements of the time for each operation, an attacker can work backwards to the input

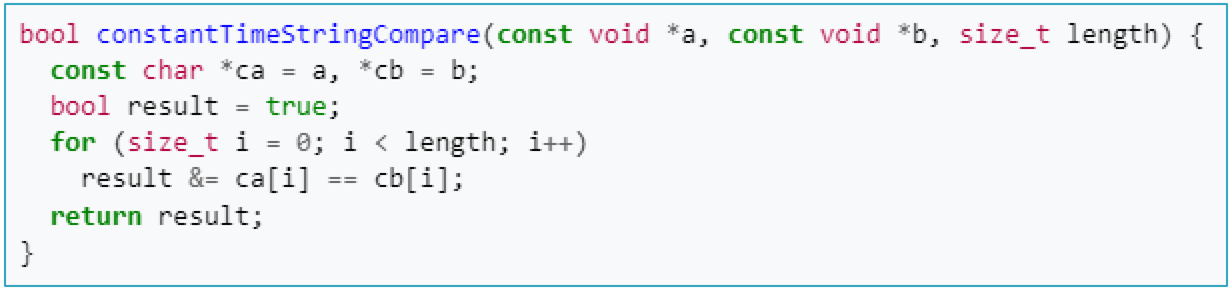
Example: Insecure String Comparison (Wikipedia)

- Spot the difference?



```c
bool insecureStringCompare(const void *a, const void *b, size_t length) {
  const char *ca = a, *cb = b;
  for (size_t i = 0; i < length; i++)
    if (ca[i] != cb[i])
      return false;
  return true;
}
```

versus



```c
bool constantTimeStringCompare(const void *a, const void *b, size_t length) {
  const char *ca = a, *cb = b;
  bool result = true;
  for (size_t i = 0; i < length; i++)
    result &= ca[i] == cb[i];
  return result;
}
```

- Note that many such functions in normal (rather than crypto-) libraries are unsafe
  - Example memcpy() as used in C

Timing Attacks

- In principal, timing attacks can be performed
  - remotely (e.g. a client measures the response time of a server that encrypts a message)
  - locally (i.e. in the host machine itself)
- Remote timing attacks are not practical, as variable OS and network latencies effect any measurement
- Local attacks are better, but require the exploit to be installed on the host under attack
- Saying this, many modern CPUs have built-in hardware instructions for AES, which protect against timing-related side-channel attacks

FYI: More Side-Channel Attacks

- **Transient execution CPU vulnerabilities** are vulnerabilities in a computer system in which a speculative execution optimisation implemented in a microprocessor is exploited to leak secret data to an unauthorized party
  - Example Meltdown and Spectre attack
- In **cache timing attacks** an attacker process deliberately causes page faults and/or cache misses in the target process, and monitors the resulting changes in access times
  - This can be done despite both processes being otherwise isolated

CT437
COMPUTER SECURITY AND FORENSIC COMPUTING

STREAM CIPHERS

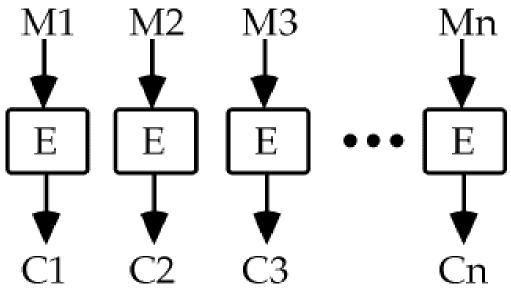
Dr. Michael Schukat

Lecture Overview

- This slide decks covers the following topics:
  - Stream Ciphers and their implementation in
    - LFSR
    - NLFSR
    - RC4
  - Pseudorandom number generation principles

Recap: Block Ciphers versus Stream Ciphers

- In a block cipher the data (e.g. text, video, or a network packet) to be encrypted is broken into blocks M1, M2, etc. of K bits length, each of which is then encrypted
- The encryption process is like a substitution on very big characters – 64 bits or more



encoding

decoding

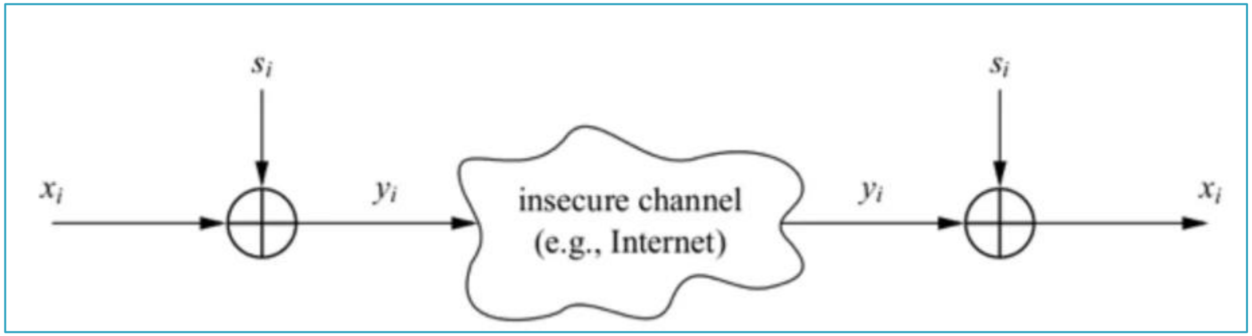
- In contrast, a **stream cipher** is a symmetric key cipher where plaintext digits are combined with a pseudorandom cipher digit stream (the **keystream**)
- Normally,
  - stream ciphers only process one bit or one byte at a time
  - the combining operation is an exclusive-or (XOR)

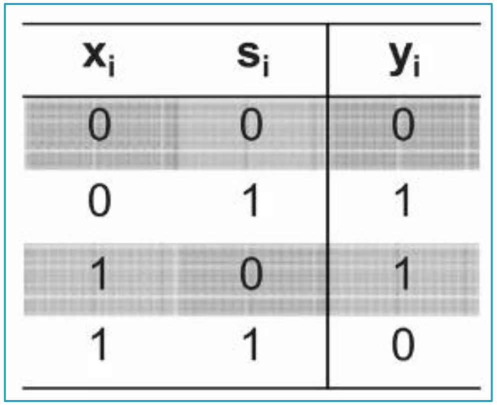
Stream Ciphers

- Stream ciphers typically provide a (pseudo) random stream key generator that produces a pseudo-random digit sequence s_i (i = 1, 2, …)
- This stream is XORed digit-by-digit with the plaintext x:

$$y_i = x_i \text{ XOR } s_i$$

- The plaintext stream can be recovered by reapplying the XOR operation
- In modern stream ciphers, a digit is one bit (or one byte → later)
- A random stream key completely destroys any statistically properties in the plaintext message
  - For a perfectly random keystream s_i, each y_i has a 50% chance of being 0 or 1
- But how can a pseudo-random sequence s_i be generated?

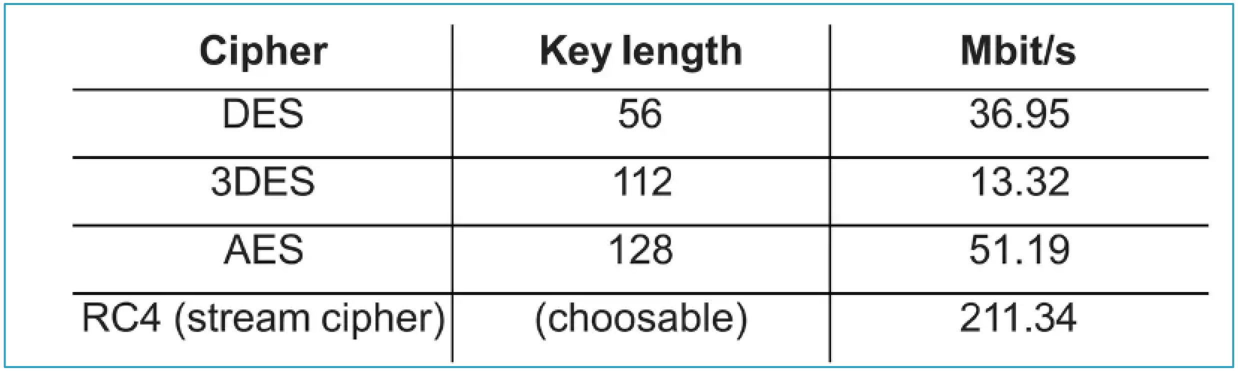




x_i	s_i	y_i
0	0	0
0	1	1
1	0	1
1	1	0

Stream Cipher Performance

- Since an XOR operation of a single bit or byte can be done in a single CPU cycle,
  - the code size and complexity of a stream cipher mainly depends on the code size and complexity of the random number generator
  - the speed of a stream cipher mainly depends on the speed of the random number generator
- For comparison (based on some Intel Pentium architecture):



Cipher	Key length	Mbit/s
DES	56	36.95
3DES	112	13.32
AES	128	51.19
RC4 (stream cipher)	(choosable)	211.34

- Size and speed make stream ciphers very suitable for resource constrained devices (e.g., mobile phones, IoT devices)

One-Time Pad

- The OTP is an encryption requires the use of a single-use pre-shared key that is equal to the size of the message being encrypted
- For the resulting ciphertext to be impossible to decrypt, the key must…
  - be at least as long as the plaintext (think of Vigenère and its weakness)
  - be
    - random (uniformly distributed in the set of all possible keys and independent of the plaintext)
    - entirely sampled from a non-algorithmic, chaotic source such as a hardware random number generator
    - pattern-less
  - never be reused in whole or in part (Coincidence counting -> next slide)
  - be kept completely secret by the communicating parties

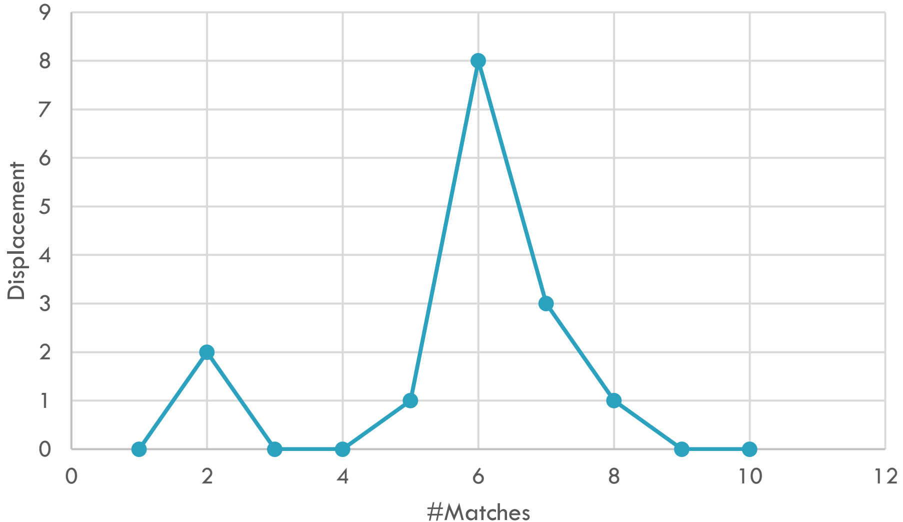
- OTPs are not practical for practical reasons, therefore pseudo-random generators (PRG) are used
- PRGs are often based on Linear Feedback Shift Registers (LFSRs)

Example Coincidence Counting

- Coincidence counting allows predicting the length of the key of a stream cipher, by comparing the ciphertext against itself with different offsets
- Assume ciphertext CXEKCWCOZKUCAYZEKW that has been encoded using a stream cipher with an unknown key
- Count the number of identical characters (matches) using different displacements of ciphertext:
  - Displacement = 1
    ```
    CXEKCWCOZKUCAYZEKW
     CXEKCWCOZKUCAYZEKW
    ```
    Matches: 0
  - Displacement = 2
    ```
    CXEKCWCOZKUCAYZEKW
      CXEKCWCOZKUCAYZEKW
    ```
    Matches: 1
  - Displacement = 3
    ```
    CXEKCWCOZKUCAYZEKW
       CXEKCWCOZKUCAYZEKW
    ```
    Matches: 0
  - …

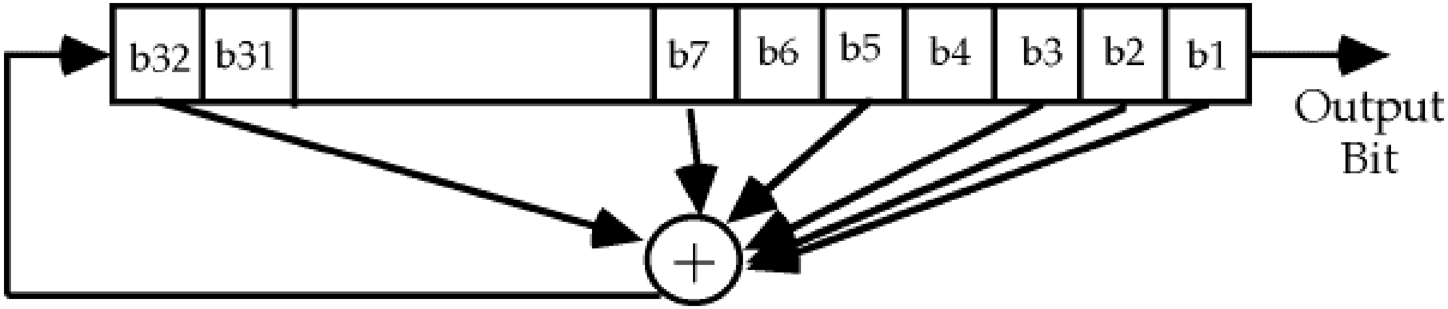
Example Coincidence Counting

- If you line up the ciphertext with itself displaced by k (= key length) characters, then you get a match in the ciphertext (offset by k places) if there is a match in the plaintext (offset by k places)
  - With the non-uniformity of the frequency distribution of English letters there's about a 6% chance that those two positions have the same letter (the index of coincidence)
- In contrast, when you line up the ciphertext using a different displacement, the index of coincidence is much smaller, i.e., 1/256, if ciphertexts are bytes
- By counting the displacement over a long ciphertext stream, k can be determined



Linear Feedback Shift Registers (LFSR)

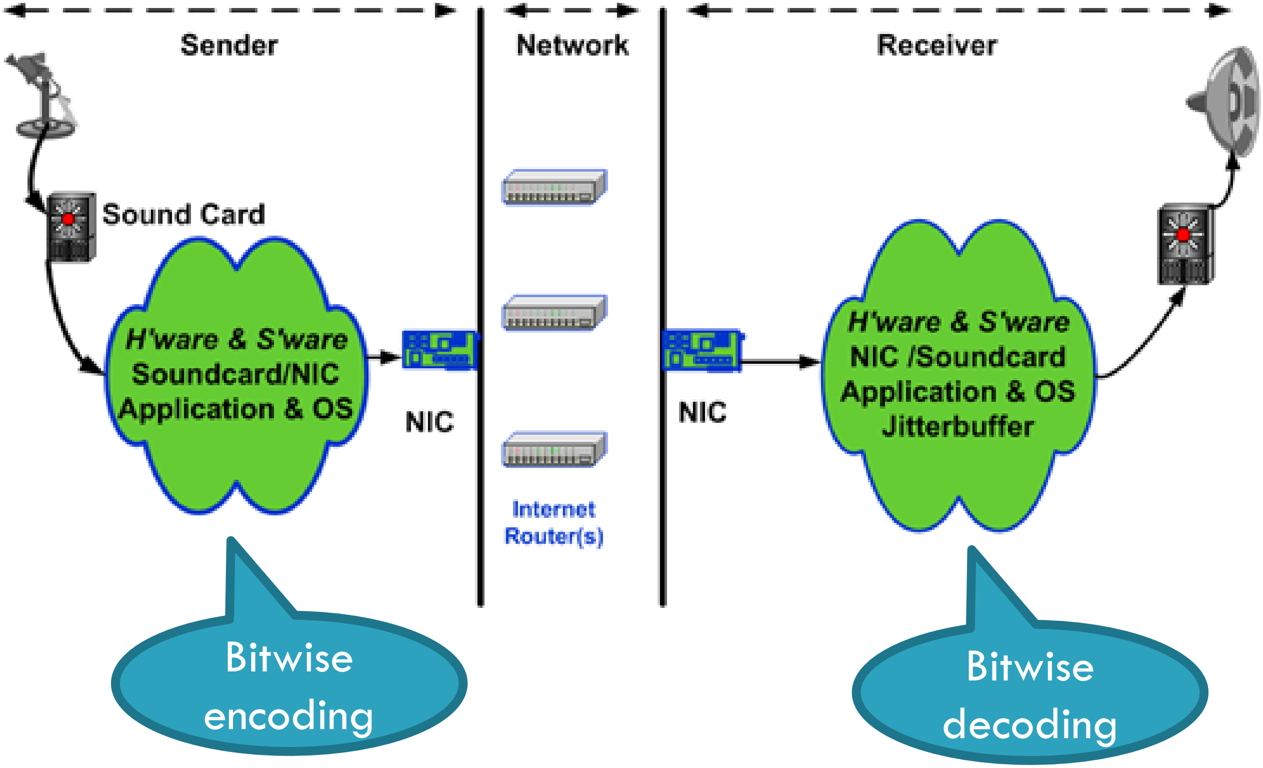
- A LFSR consists of a binary shift register of some length along with a linear feedback function (LFF) that operates on some of those bits
  - The most commonly used LFF is the XOR operation
- To get started the register is preset with a secret initialisation vector
- Each time a bit is needed,
  - a new bit is formed from the linear feedback function
  - all bits are shifted by one position (shifted right in the example below) with the new bit being shifted in
- The bit shifted out is used as the (pseudo-random) output of the LFSR
- A well-designed n-bit LFSR generates a pseudo-random sequence whose length correlates to n



Example for an 8-Bit LFSR

- □ Initialisation vector: $\underline{1}01\underline{0}01\underline{1}0 \quad (B_7 \cdots B_0)$
- □ Feedback Function: $B_7 \text{ XOR } B_4 \text{ XOR } B_1$
- □ Right shift after each cycle (B_0 shifted out)
- □ Iteration 0: 10100110
- □ Iteration 1: $\underline{0}10\underline{1}00\underline{1}1 \quad >> \ 0$
- □ Iteration 2: $\underline{0}01\underline{0}10\underline{0}1 \quad >> \ 1$
- □ Iteration 3: $\underline{0}00\underline{1}01\underline{0}0 \quad >> \ 1$
- □ Iteration 4: $10001010 \quad >> \ 0$
- □ … …

Example VoIP Encoding using a Stream Cipher



Bitwise encoding

Bitwise decoding

Stream Ciphers in Practice

- In practice, one key is used to encrypt many messages
  - Example: Wireless communication
  - Solution: Use Initial vectors (IV)
  - $E_{key}[M] = [IV, M \oplus PRNG(key \mid\mid IV)]$
    - IV is sent in clear to receiver
    - IV needs integrity protection, but not confidentiality protection
    - IV ensures that key streams do not repeat, but does not increase cost of brute-force attacks
    - Without key, knowing IV still cannot decrypt
  - Need to ensure that IV never repeats! How?

Example for a 16-bit LFSR written in C

```
#include <stdint.h>
#include <stdio.h>
int main(void) {
    uint16_t start_state = 0xACE1u;  /* Any non-zero start state will work. */
    uint16_t lfsr = start_state;
    uint16_t bit, input, period = 0;
    printf(“Enter LFSR IV as integer: ”); scanf(“%d”, &input);
    if (input > 0) {
       start_state = input;
       lfsr = start_state;
    }
    do
    {   /* LFF: B15 XOR B13 XOR B12 XOR B10 */
        bit = ((lfsr >> 0) ^ (lfsr >> 2) ^ (lfsr >> 3) ^ (lfsr >> 5)) & 1u;
        lfsr = (lfsr >> 1) | (bit << 15);
        printf(“%d”, bit);
        ++period;
    } while (lfsr != start_state);
    printf(“\nPeriod of output sequence: %d \n”, period);
    return 0;
}
```

What is the Maximum Sequence Length of a single LFSR?

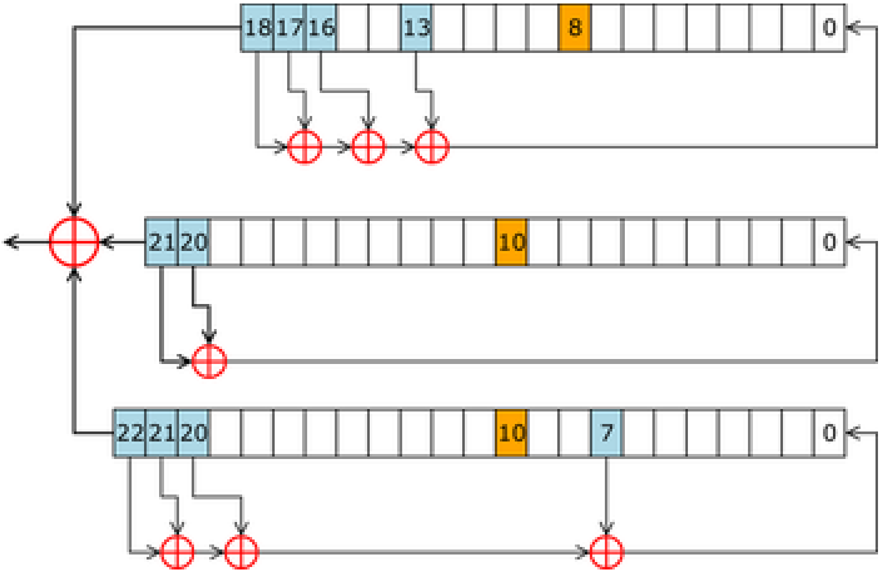
- Consider a single n-bit LFSR with some feedback function
- Each bit that is shifted out is intrinsically linked to the content of the LFSR
- Each shift operation maps the register content to another (different) pattern, as seen in the example, resulting in another bit shifted out
- An n-bit LFSR allows for 2^n different register content variations, with each variation pushing out a 0 or a 1
- Therefore, the longest cycle of non-repeating patterns is $2^n – 1$ iterations, with 2^n the maximum length of the sequence
  - Think of a 1-bit LFSR (n = 1):
    - There are 2 different LFSR contents (“0” or “1”) possible
    - The longest possible patterns are “10” or “01”; both have a length of 2^n
    - It just takes one iteration (2^{n-1}) to reach all possible register contents (1 → 0 or 0 → 1)
- However,
  - poorly designed LFSR may result in cycles that are shorter
  - the Index of Coincidence problem also applies to LFSR (and in fact to all stream ciphers)

The Combined LFSR

- A combined LFSR uses multiple LFSR in parallel, and combines their respective outputs to generate a key stream
- They work well on resource-constrained devices too
- Example: A5/1, which was used for GSM voice communication:
  - The Global System for Mobile Communications (GSM) was a mobile phone standard back in the 1990s
  - In GSM, digitised phone conversations are sent as sequences of frames
  - One frame is sent every 4.6 milliseconds and is 228 bits in length
    - Voice samples are collected / digitised over 4.6 milliseconds and send in a block
  - A5/1 is a combined LFSR-based algorithm that is used to produce 228 bits of key stream which is XORed with the frame
  - It is initialised using a 64-bit key

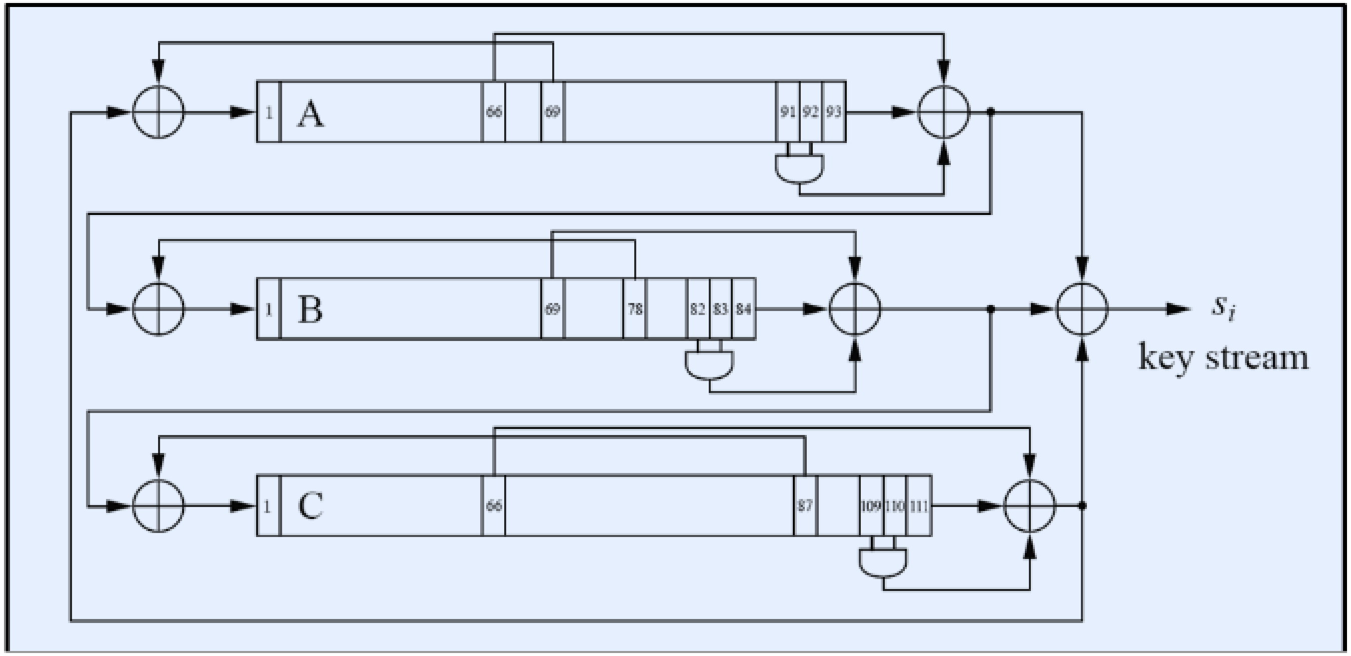
Example A5/1

- 3 independent LFSRs:
  - LFSR 1
    - 19 bits
    - LFF: B18 XOR B17 XOR B16 XOR B13
  - LFSR 2:
    - 22 bits
    - LFF: B21 XOR B20
  - LFSR 3:
    - 23 bits
    - LFF: B22 XOR B21 XOR B20 XOR B7
- The output bit is the XORed output of all 3 LFSRs
- A LFSR is only shifted to the left, if their clocking bit (B8, B10, and B10 respectively) matches the output bit;
  otherwise, there is no shift, and the same output bit value is used again in the next cycle



Non-Linear Feedback Shift Registers (NLFSR)

- NLFSR contain AND gates as well as XOR gates in their feedback function
- Example Trivium: A, B and C are three shift registers with bit lengths of 93, 84 and 111 bits respectively



Example for a 16-bit NLFSR in C

```
#include <stdint.h>
#include <stdio.h>
int main(void)
{
    uint16_t start_state = 0xACE1u;  /* Any non-zero start state will work. */
    uint16_t lfsr = start_state;
    uint16_t bit, period = 0;
    do
    {   /* FBF: B15 XOR B13 XOR B12 XOR B10 XOR (B2 and B1)*/
        bit = ((lfsr >> 0) ^ (lfsr >> 2) ^ (lfsr >> 3) ^ (lfsr >> 5) ^ ((lfsr >> 13) & (lfsr >> 14))) & 1u;
        lfsr = (lfsr >> 1) | (bit << 15);
        printf(“%d”, bit)
        ++period;
    } while (lfsr != start_state);
    printf(“\nPeriod of output sequence: %d \n”, period);
    return 0;
}
```

Pseudo-Random Number generation: RC4

- Instead of single bits, a generator algorithm can also produce one byte (or one word) at a time
- RC4 is an example for such an algorithm, it returns one pseudorandom byte at a time
- It was designed by Ron Rivest of RSA Security in 1987
- RC4 was initially a trade secret, but in 1994 a description of it was anonymously posted on the Internet
- RC4 consists of a
  - key-scheduling algorithm (KSA) and a
  - pseudo-random generation algorithm (PRGA)

RC4: The Key-Scheduling Algorithm (KSA)

- The KSA requires a key (stored in key[]) of length keylength
  - keylength is somewhere between 1 and 256
- Using the keyword, a 256-byte long permutation vector S[] is generated:

```
for i from 0 to 255
    S[i] := i;
j := 0;
for i from 0 to 255
    j := (j + S[i] + key[i mod keylength])
              mod 256;
    swap(S[i],S[j]);
```

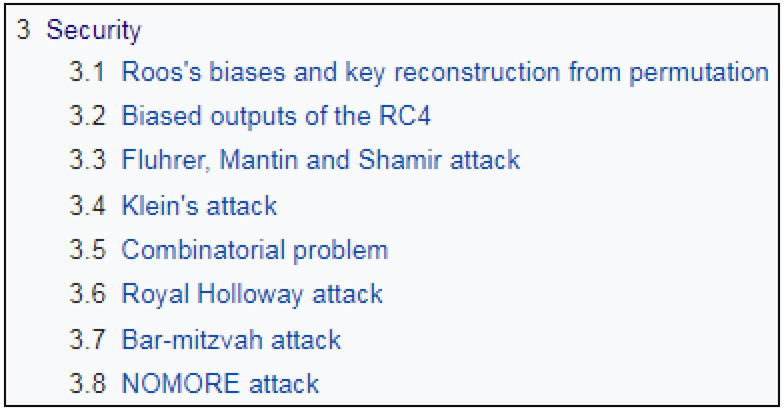
RC4: The Pseudo-Random Generation Algorithm (PRGA)

- PRGA returns one byte at a time:

```
i := 0;
j := 0;
while GeneratingOutput:
    i := (i + 1) mod 256;
    j := (j + S[i]) mod 256;
    swap(S[i],S[j]);
    output S[(S[i] + S[j]) mod 256];
```

Security of RC4

- Obviously not an LFSR-based design, but a more general pseudo-random number generator design
- Can also be efficiently implemented in software
  - Very compact algorithm
- However, it is not deemed safe anymore!



3 Security
- 3.1 Roos's biases and key reconstruction from permutation
- 3.2 Biased outputs of the RC4
- 3.3 Fluhrer, Mantin and Shamir attack
- 3.4 Klein's attack
- 3.5 Combinatorial problem
- 3.6 Royal Holloway attack
- 3.7 Bar-mitzvah attack
- 3.8 NOMORE attack

Background: Pseudorandom Number Generators

- Cryptographically strong pseudorandom number generation is essential!

TOUR OF ACCOUNTING
OVER HERE WE HAVE OUR RANDOM NUMBER GENERATOR.
scottadams@aol.com
www.dilbert.com
NINE NINE NINE NINE NINE NINE
© 2001 United Feature Syndicate, Inc.
ARE YOU SURE THAT'S RANDOM?
THAT'S THE PROBLEM WITH RANDOMNESS: YOU CAN NEVER BE SURE.

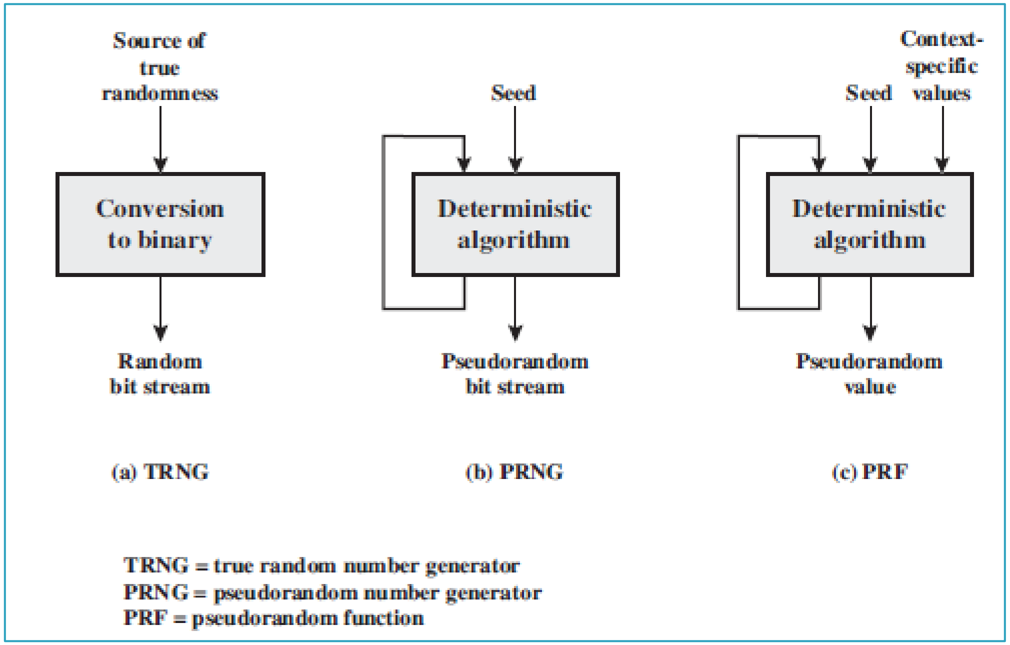
- Pseudorandom number generators (PRNG) are used in a variety of cryptographic and security applications, including
  - Stream cipher encryption → 802.11 WEP
  - Encryption keys (both for symmetric and public key algorithms)

Obvious Requirements for Random Number Generators

- Assume we toss a fair coin or throw a fair dice multiple times. We expect the following from the resulting sequence:
- Randomness, i.e. uniform distribution
  - The distribution of values in the sequence (e.g. “head or tail”) should be uniform; that is, the frequency of occurrence of possible outputs should be approximately equal
- Unpredictability, i.e. independence
  - Successive members of the sequence are unpredictable; no subsequence in the sequence can be inferred from the others

Types of Random Generators

- A TRNG takes as input a source that is effectively random
  - The source is often referred to as an **entropy source**
  - The entropy source is drawn from the physical environment of the computer, e.g. a combination of keystroke timing patterns, CPU temperature changes and mouse movements



TRNG = true random number generator
PRNG = pseudorandom number generator
PRF = pseudorandom function

- A PRNG uses just a seed (e.g. LFSR)
- A PRF often also takes in a context-specific value, e.g.
  - A secure end-to-end communication via TCP/IP may take in the endpoints' IP addresses
- However, PRNG and PRF are based on deterministic algorithms, therefore the "P"

Formal Requirements for Pseudorandom Generators

- Randomness
  The generated bit stream must “appear” random even though it is deterministic
  This can be validated by applying a sequence of tests to the generator, which determine (among others) the following characteristics:
  - Uniformity: At any point in the generation of a sequence of random or pseudorandom bits, the occurrence of a zero or one is equally likely; The expected number of zeros (or ones) is n/2, with n being the sequence length
  - Scalability: Any test applicable to a sequence can also be applied to sub-sequences extracted at random; if a sequence is random, then any such extracted subsequence should also be random
  - Consistency: The behavior of a generator must be consistent across many starting values (seeds); it is inadequate to test a PRNG based on the output from a single seed

Formal Requirements for Pseudorandom Generators

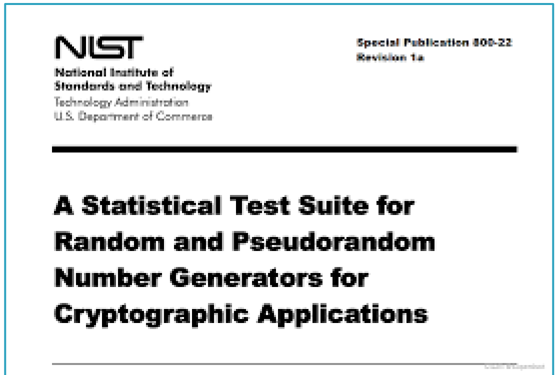
- Unpredictability

  A stream of pseudorandom numbers should exhibit two forms of unpredictability

  - Forward unpredictability: If the seed is unknown, the next output bit in the sequence should be unpredictable in spite of any knowledge of previous bits in the sequence
  - Backward unpredictability: It should not be feasible to determine the seed from knowledge of any generated values; no correlation between a seed and any value generated from that seed should be evident; each element of the sequence should appear to be the outcome of an independent random event whose probability is 0.5

NIST SP 800-22

- □ The National Institute of Standards and Technology (NIST) published the above report, "*A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*"
- □ It lists 15 separate tests of randomness and unpredictability
- □ https://github.com/terrillmoore/NIST-Statistical-Test-Suite



NIST
National Institute of Standards and Technology
Technology Administration
U.S. Department of Commerce

Special Publication 800-22
Revision 1a

A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications

CT5191
CRYPTOGRAPHY AND NETWORK SECURITY

HASH FUNCTIONS AND MESSAGE AUTHENTICATION CODES

Dr. Michael Schukat

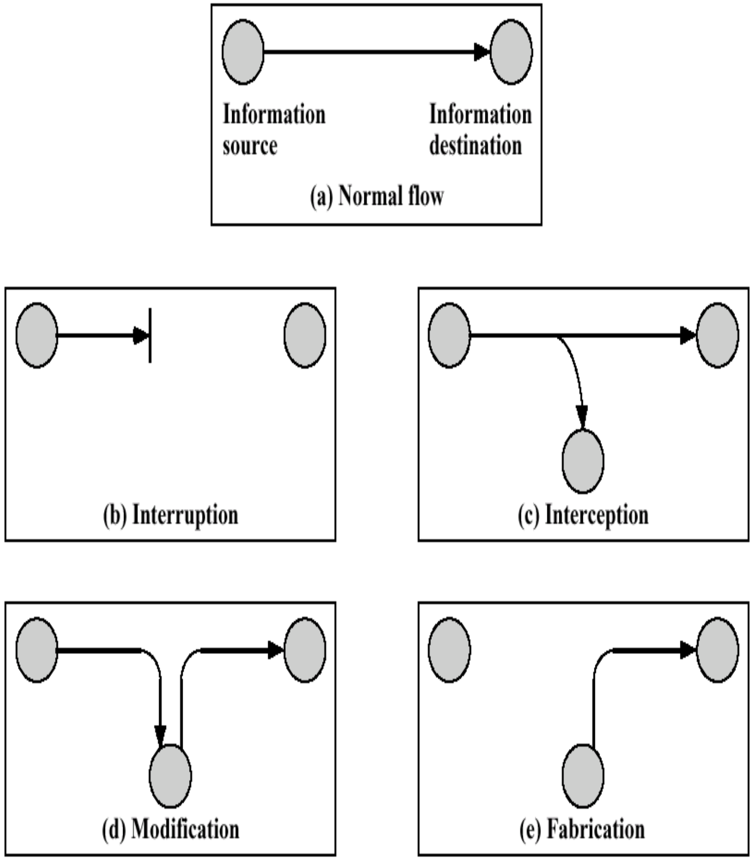
OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Lecture Overview

- In the previous lectures we have covered block and stream ciphers that provide data confidentiality
- In this slide deck we focus on data integrity, i.e., "*Guarding against improper information modification or destruction, and includes ensuring information non-repudiation and authenticity*"
- Such integrity protection can be provided via
  - Message authentication codes
  - Hash functions

Recap: Types of Security Attacks on Information in Transit

- Integrity checks are particularly important for data in transit
- Here we need to consider the following **active** and **passive** attacks:
  - **Interception** - of info-traffic flow, attacks confidentiality
  - **Interruption** - of service, attacks availability
  - **Modification** - of info, attacks integrity
  - **Fabrication** - of info, attacks authentication
- In all these scenarios the attacker is a "Man-in-the-Middle" (MitM)



(a) Normal flow

(b) Interruption

(c) Interception

(d) Modification

(e) Fabrication

Recap: Passive Attacks

- Passive attacks are in the nature of eavesdropping or the monitoring of transmissions:
  - Release of plaintext message content
  - Traffic analysis of encrypted data communication
    - Allows to analyse patterns of message exchange (sender, receiver, timing) rather than content
- Tools like Wireshark allow for passive attacks

Recap: Active Attacks

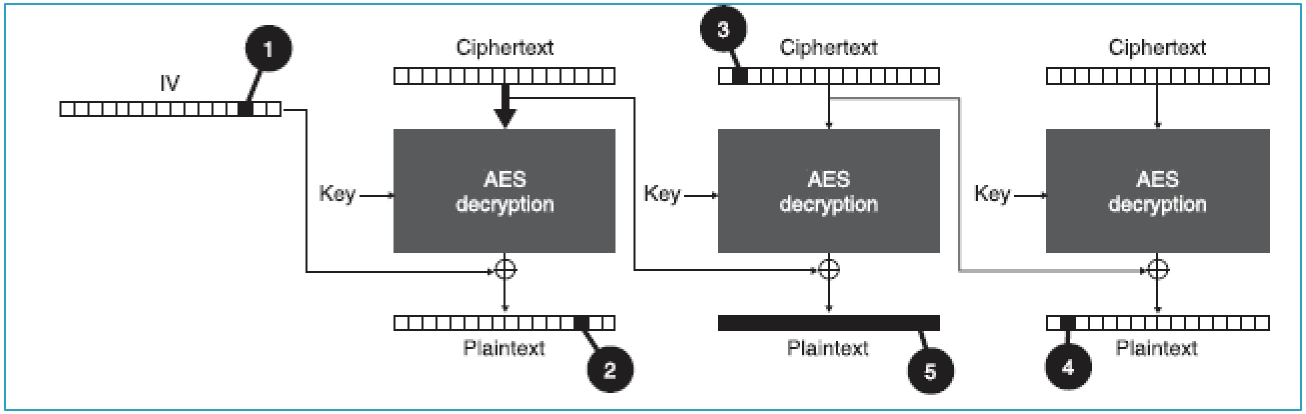
- Active attacks involve the modification or the creation of data in a stream:
  - **Masquerade**
    - **Attacker pretends to be a legitimate sender or receiver of data**
  - Replay
    - Attacker retransmits (encrypted) data which has been previously captured via eavesdropping
  - **Modification of message content**
    - **Attacker intercepts a message in transit, modifies it and forwards it to the receiver**
  - Denial of Service (DoS)
    - Attacker Inhibits the normal use of communication services

Attack Scenario

- Your company sends the software patch as email attachment to all the clients
- The patch is encrypted using a secret key, which is pairwise shared with your clients
- However, an attacker can
  - intercept these emails in transit, changes randomly a few bytes of the encrypted executable and forwards them to their destination, or
  - forge a similar looking email with some random file attached that claims to be a bug fix
- Your clients replace the executable on their local machines, which of course won’t work and bring the entire factory floor to a halt
  - → financial losses for your clients, huge reputational loss for your company!
- **Therefore, your clients need some mechanism to validate the origin of the email, as well as the integrity of its content**

Case Study 2: Weakness of Mode Block Cipher Modes

- In CBC, the IV is tagged to an encrypted message as plaintext (thereby allowing the receiver to decrypt the message), a MitM attacker can do changes in transit. Here:
  - Flipping the i^th IV-bit (1) flips also the i^th plaintext bit (2)
  - Flipping a ciphertext bit (3) will change the entire plaintext block (5), and the corresponding bit of the next plaintext block (4)
- Other modes show similar weaknesses, i.e. changing one bit in a single block of an encrypted message (in transit) will corrupt the correct decoding of a following blocks
- **The receiver needs the ability to validate the integrity of the received message (blocks) !**

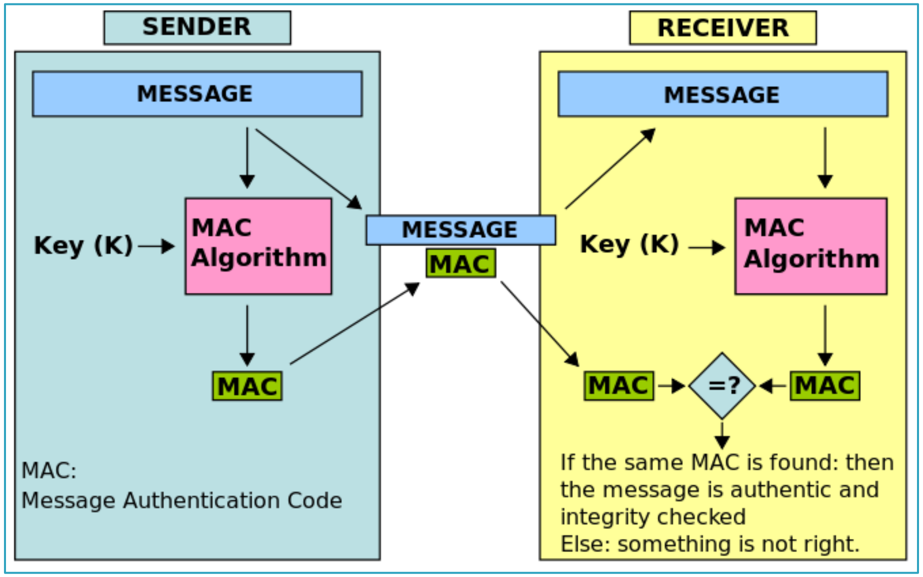


Message Authentication Code (MAC)

- Message authentication = message integrity [+ source authentication]
- A MAC (also called authentication tag, fingerprint, or cryptographic checksum), is a short piece of information used for authenticating and integrity-checking a message
- A MAC condenses a variable-length message M using a secret key K and some algorithm C to a fixed-sized authenticator: MAC = $C_K(M)$
- After its calculation, the MAC is appended to the message before it is sent
- Note that the message:
  - can have any length
  - is not automatically encrypted!

Typical Use of a MAC (Wikipedia)

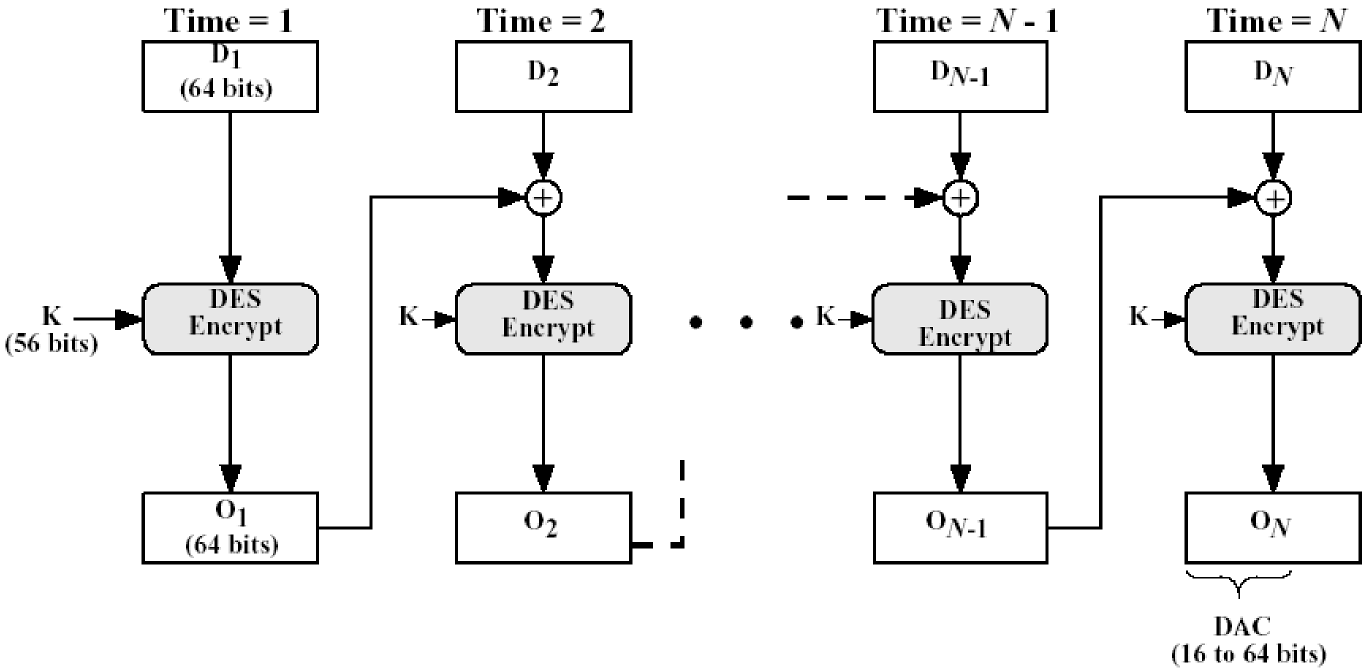
- If both MACs are identical, the receiver knows, that
  - the message was not altered in transit,
  - the message was sent by the alleged sender, and
  - if the message includes a sequence number, that the sequence was not altered
- The term **CMAC** is used for a MAC that is calculated using a (block) cipher
- This contrasts to a **HMAC**, where a hash function (later) and a secret key is used



MAC:
Message Authentication Code

If the same MAC is found: then the message is authentic and integrity checked
Else: something is not right.

Typical CMAC Implementation



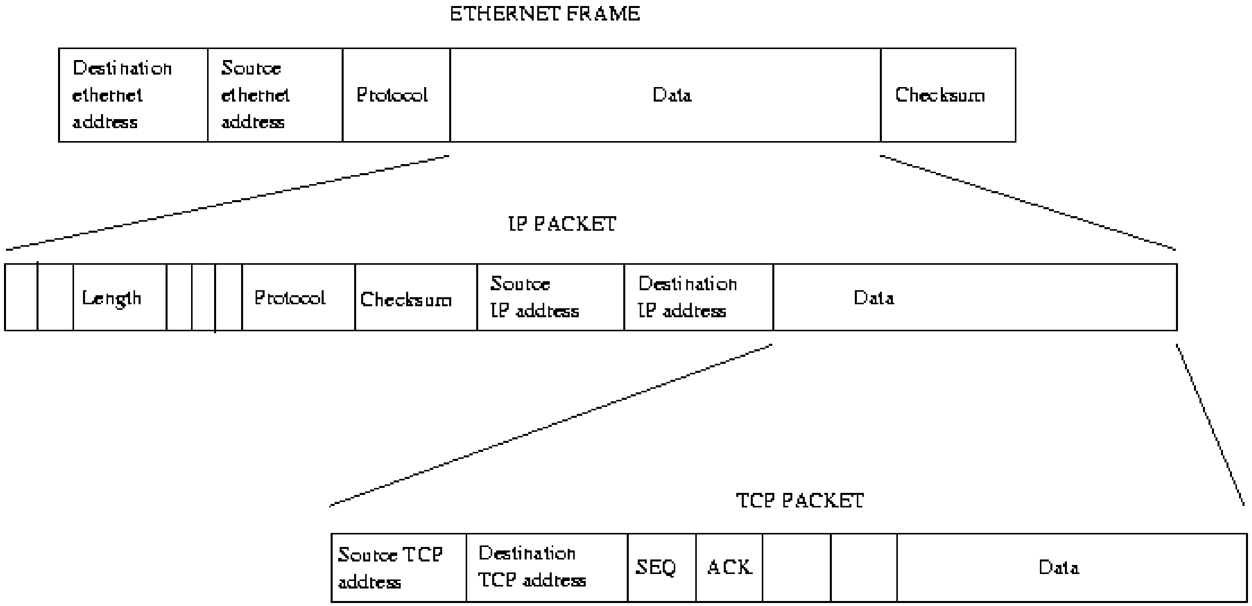
- Generally:
  - Any modern block cipher may be used (i.e., it's only DES in the example above)
  - Message padding shall apply as seen before
  - MAC = $C_K(M)$, where K is secret key and C is a symmetric block cipher (DES above)
  - MAC guarantees message integrity AND source authentication
  - This construction is also called **Encrypt-then-MAC**

Message Authentication Benefits

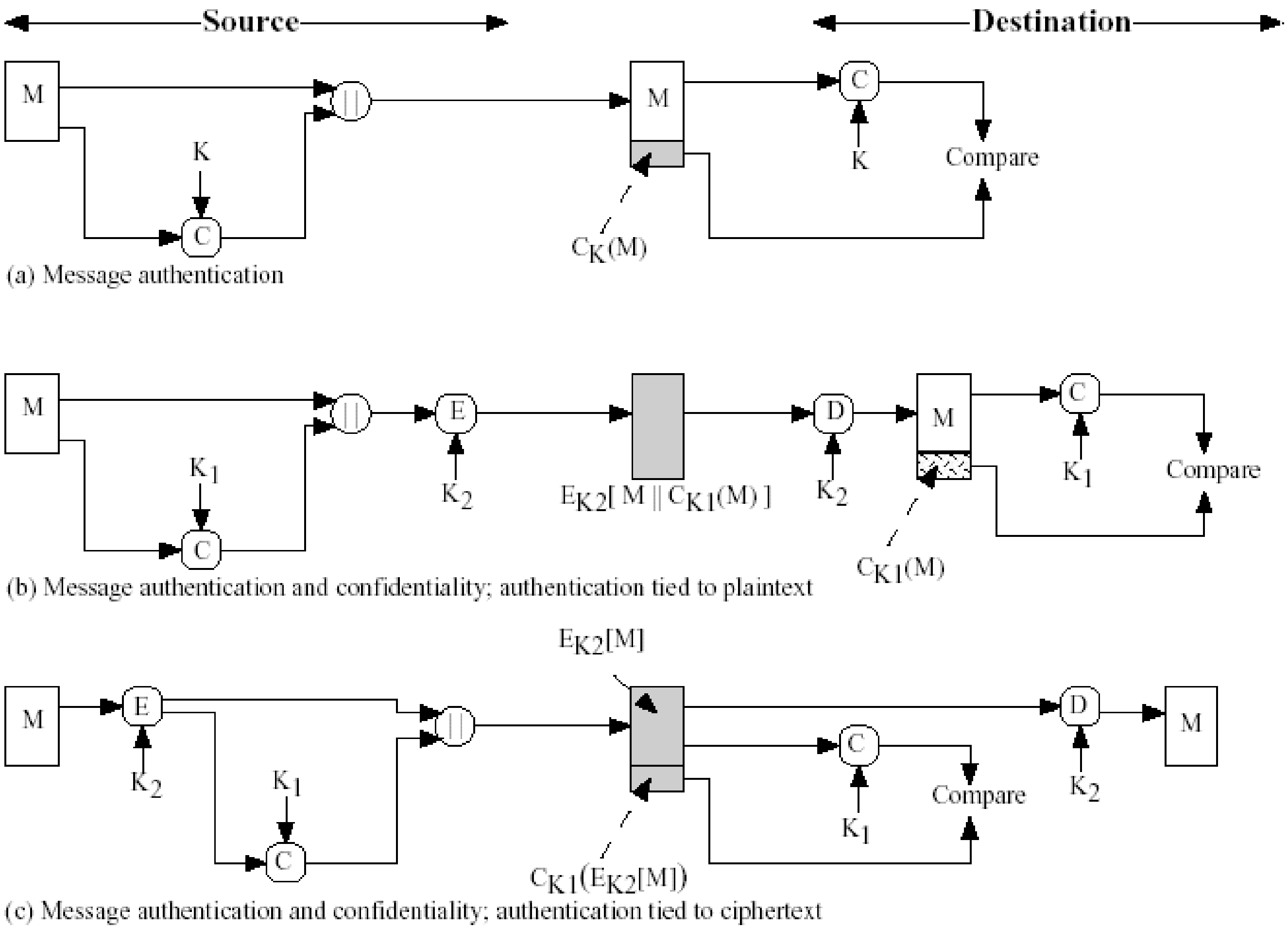
- In summary there are four types of attacks on data in transit, which are addressed by message authentication:
  - **Masquerade:** insertion of messages into the network from a fraudulent source
  - **Content modification**
  - **Sequence modification:** change the order of messages as they arrive
  - **Timing modification:** delete or repeat messages
- Note that the above may require a unique (i.e. incremented) sequence number in every message
- Therefore, message authentication is concerned with:
  - Protecting the integrity of a message
  - Validating identity of originator
  - Validating sequencing and timeliness
  - Non-repudiation of origin (dispute resolution)

Example: Authentication of TCP/IP Packets

- In TCP/IP data communication, a MAC cannot only cover the payload (i.e., the TCP Data field), but also the TCP header, as well as the non-modifiable fields of the IP header



Basic Use Cases of CMACs



(a) Message authentication

(b) Message authentication and confidentiality; authentication tied to plaintext

(c) Message authentication and confidentiality; authentication tied to ciphertext

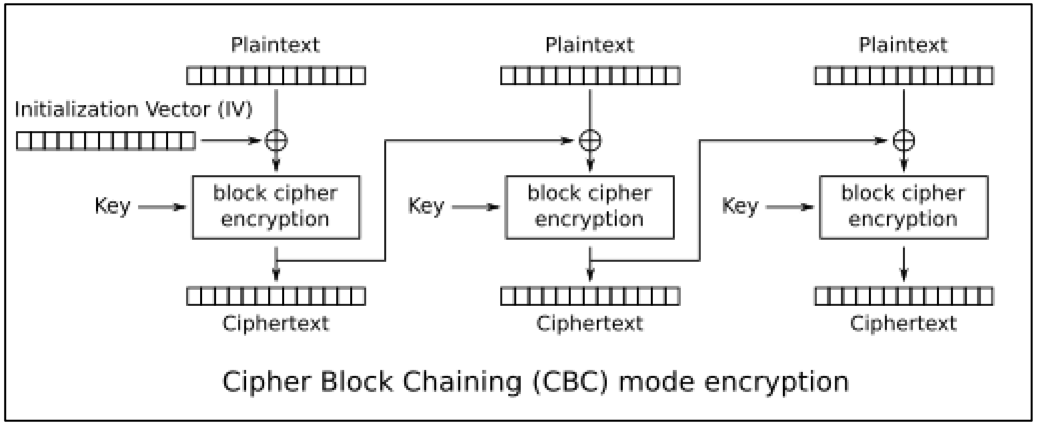
- M: Message
- K: Secret key
- C: Block cipher
- ¦¦:Concatenation operation

Case Study CMAC

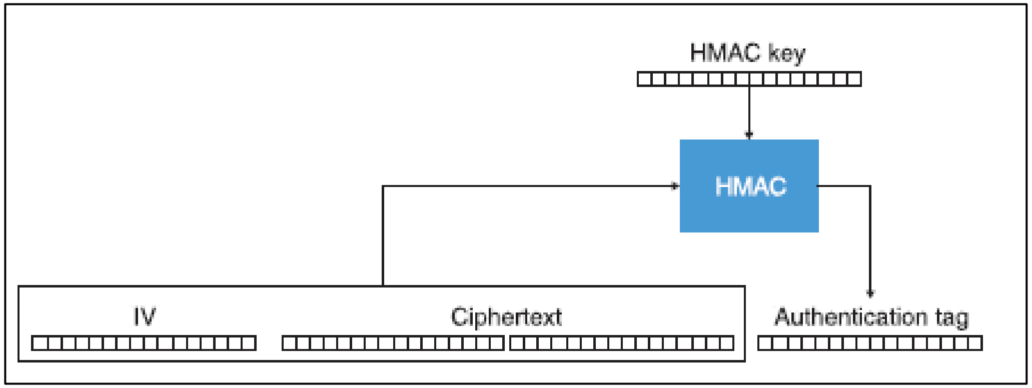
- Assume you operate a distributed weather station with battery-operated sensors located across Ireland
- You use “public” networks (i.e. Wi-Fi, Internet) to collect data and send it for processing to a central hub in Galway
- Which basic uses of a CMAC as shown in the previous slide would be most appropriate?
  - In your suggestion consider data privacy concerns and energy budget

The AES-CBC-HMAC Mode

- An example on how to combine authentication with a block cipher mode
- Based on CBC mode (top), but with additional authentication (bottom)
- Here the HMAC takes a single variable length input, i.e. the concatenation of IV + ciphertext + HMAC key, and creates a fix length authentication key
  - The diagram is misleading as it shows two separate inputs
- How many secret keys would this scheme require?

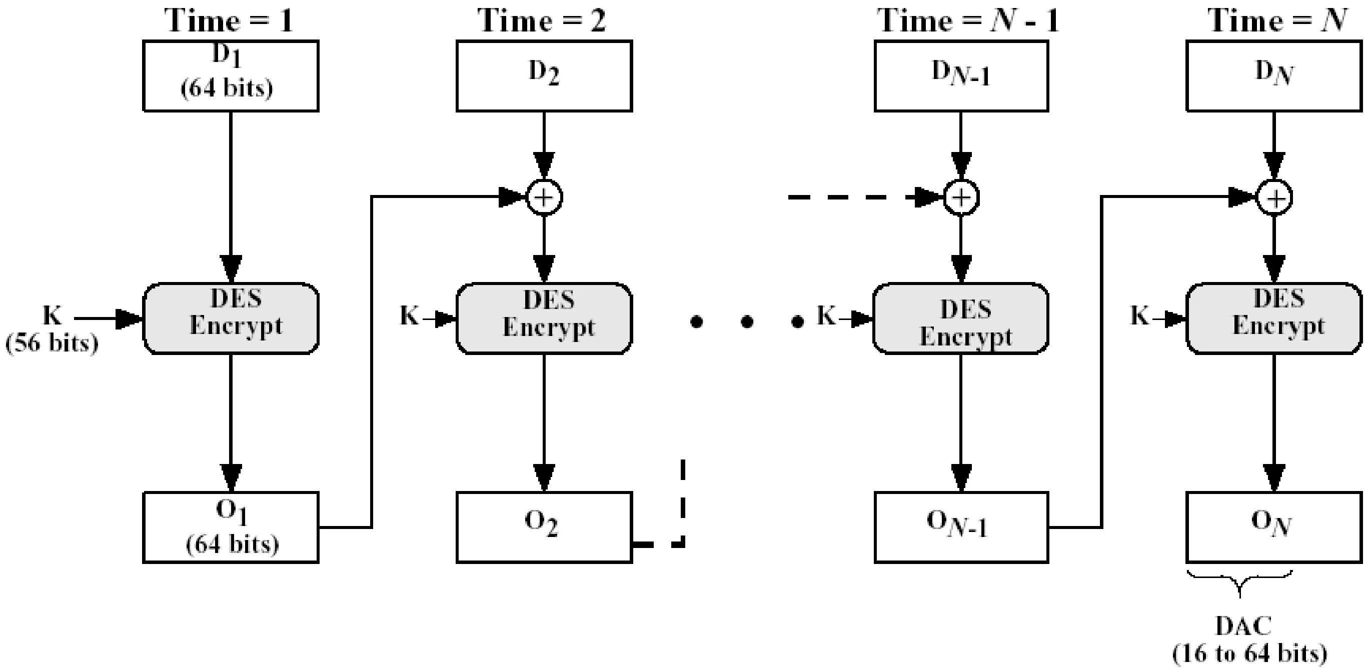


Cipher Block Chaining (CBC) mode encryption



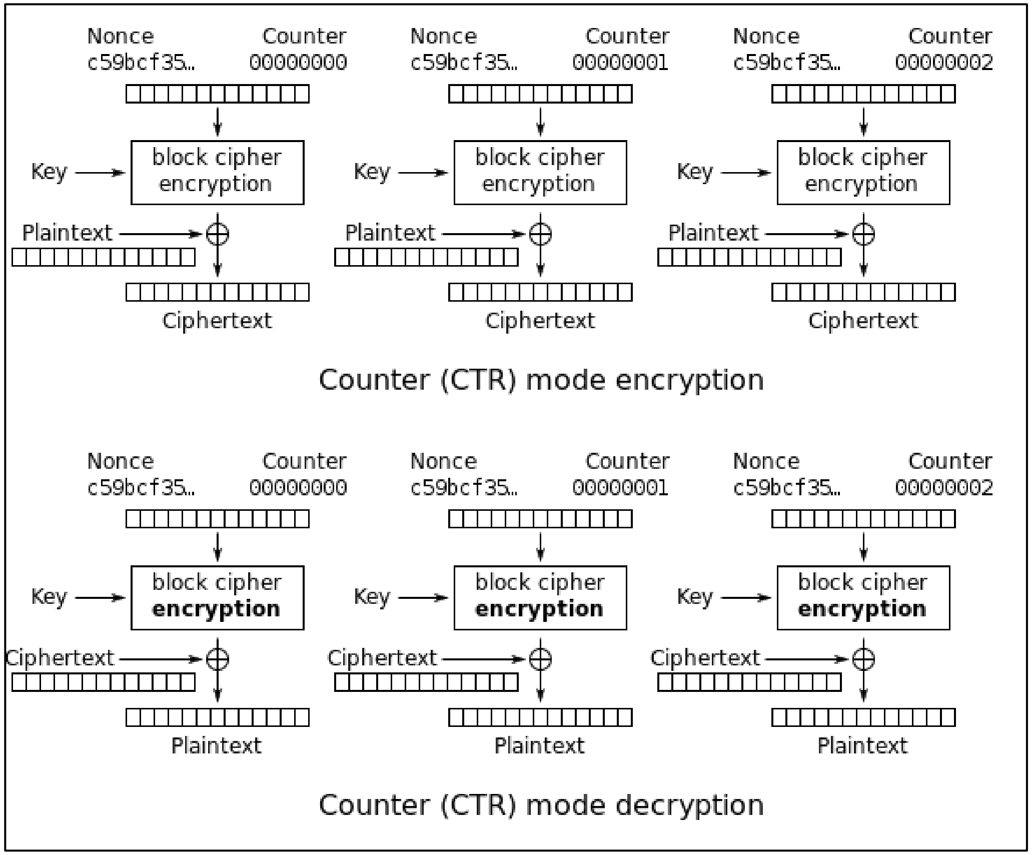
Block Cipher Mode of Operation: The Galois / Counter Mode

- What are weaknesses of the mode below and the AES-CBC-HMAC Mode (previous slide), i.e.
  - Can it be parallelised?
  - Is a 16- to 64-bit DAC sufficient?



Block Cipher Mode of Operation: The Galois / Counter Mode

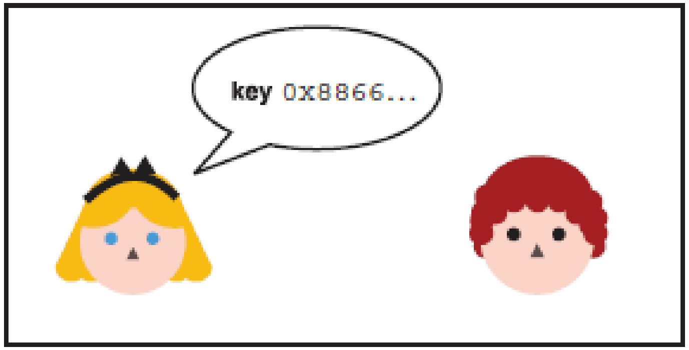
- □ Extension of counter mode
- □ Recall advantages of this mode?



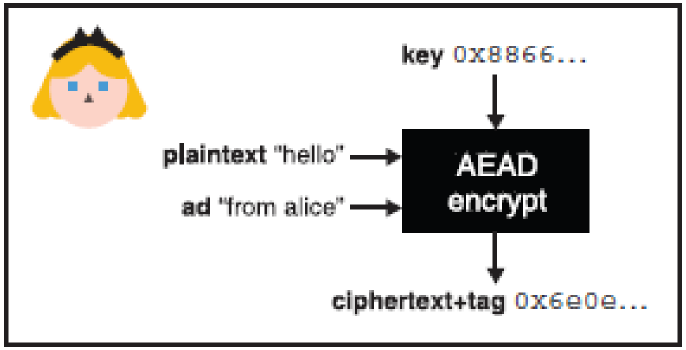
Block Cipher Mode of Operation: The Galois / Counter Mode

- GCM provides both data authenticity (integrity) and confidentiality
- It belongs to the class of **authenticated encryption with associated data (AEAD)** methods, i.e. it takes as an input
  - an initialisation vector IV
  - a **single** secret key K,
  - the plaintext P, and
  - some associated data AD
- It encrypts the plaintext (similar to counter mode) using the key to produce ciphertext C, and computes an authentication tag T from the ciphertext and the associated data (which remain unencrypted)
- A recipient with knowledge of K, upon reception of AD, C and T, can decrypt the ciphertext to recover the plaintext P and can check the tag T to ensure that neither ciphertext nor associated data were tampered with
- GCM uses a block cipher with block size 128 bits (i.e., AES-128), and uses arithmetic in the Galois field GF(2^{128}) to compute the authentication tag
  - That's modular arithmetic with a modulus of 2^{128}

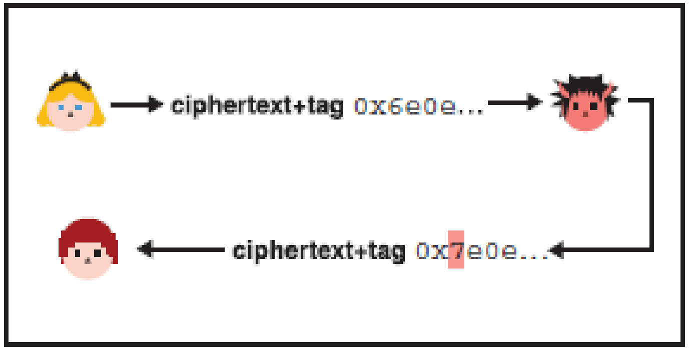
Features of AEAD



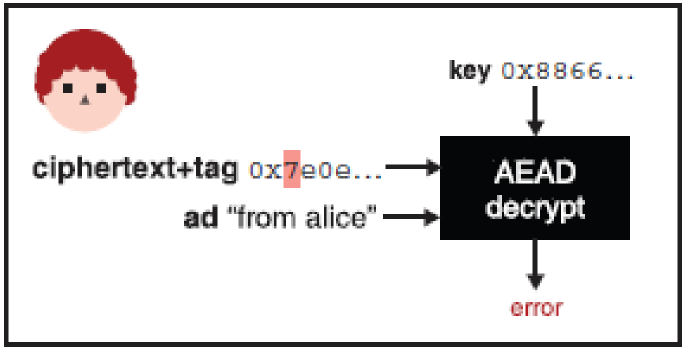
1. Alice and Bob meet in real life to **agree on a key**.



2. Alice can now use it to encrypt messages with an **AEAD algorithm** and the **symmetric key**. She can also add some optional associated data.



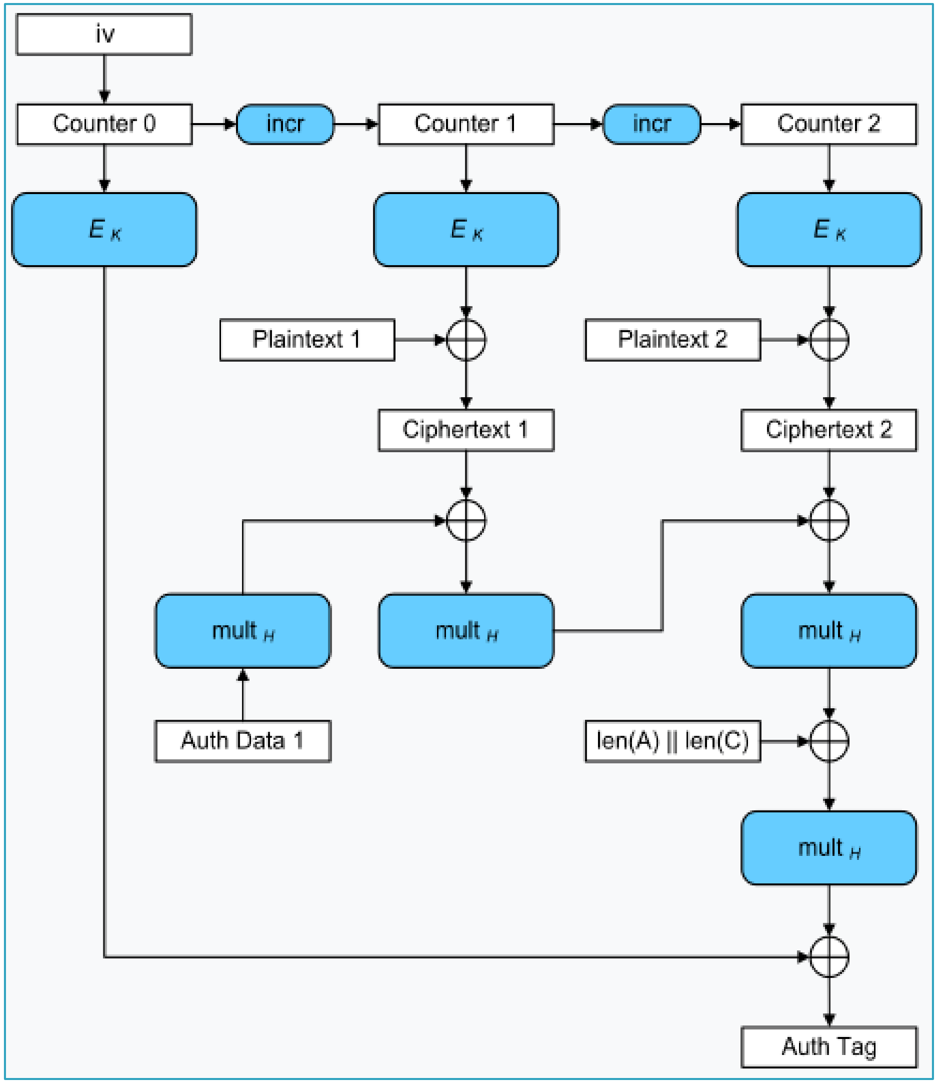
3. The ciphertext and tag are sent to Bob. An observer on the way **intercepts** them and **modifies** the ciphertext.



4. Bob uses the AEAD decryption algorithm on the **modified ciphertext** with the same key. **The decryption fails.**

Block Cipher Mode of Operation: The Galois / Counter Mode

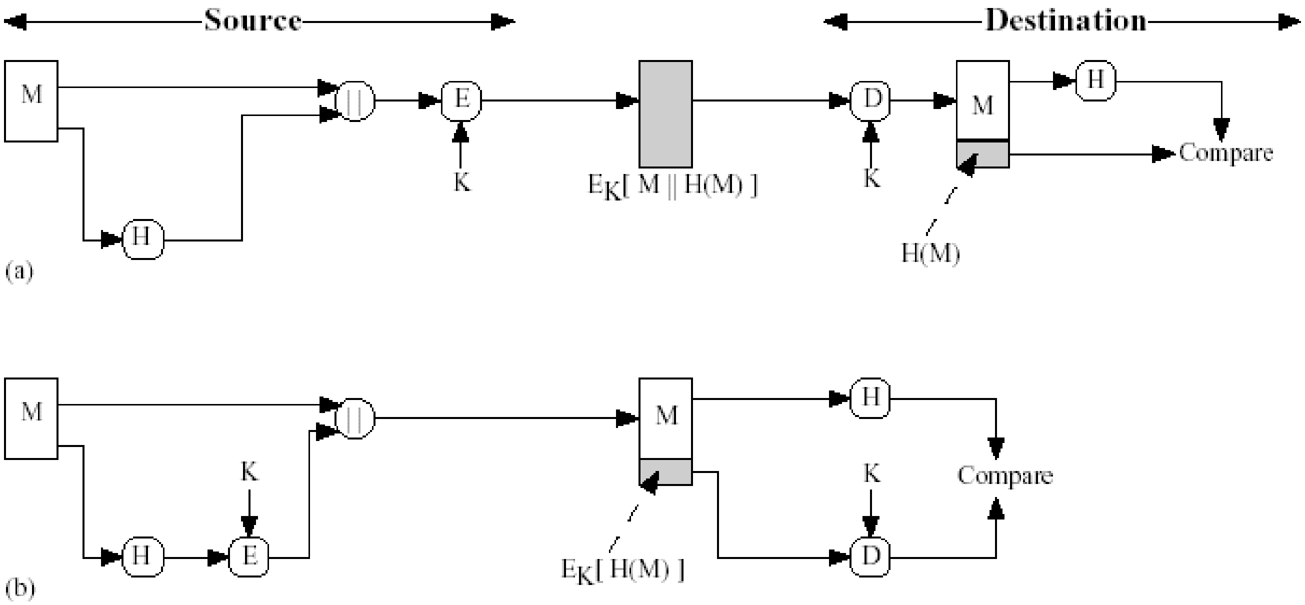
- □ A 96-bit IV is concatenated with a 32-bit counter (initialised with 0), i.e. (IV << 32) | | C
- □ E_K is AES with a 128 – 256 bit key (AES-128, AES-192 or AES-256)
- □ $mult_H$ is a hash-function (later) that produces a 128-bit (hash) output
- □ Auth_Data_1 has a variable length (but its hash is 128-bit wide)
- □ len(A) and and len(C) are 64-bit values that are the lengths (in bytes) of Auth_Data_1 and all ciphertext blocks respectively
- □ ⊕ is the bitwise XOR function



Hash Functions and HMAC

- A hash function produces a fixed size hash code (i.e. hash or fingerprint) based on a variable size input message
  - A hash function
    - does not need a key
    - guarantees the integrity of the message
- However, since a hash function is public and is not keyed, a hash value may have to be protected (i.e., encrypted)
  - A HMAC (hash-based message authentication code) is a specific type of MAC involving a cryptographic hash function and a secret cryptographic key
  - A HMAC verifies both message integrity and its authenticity
- Modern hash functions calculate 256 - 512-bit hashes

Basic Uses of HMACs



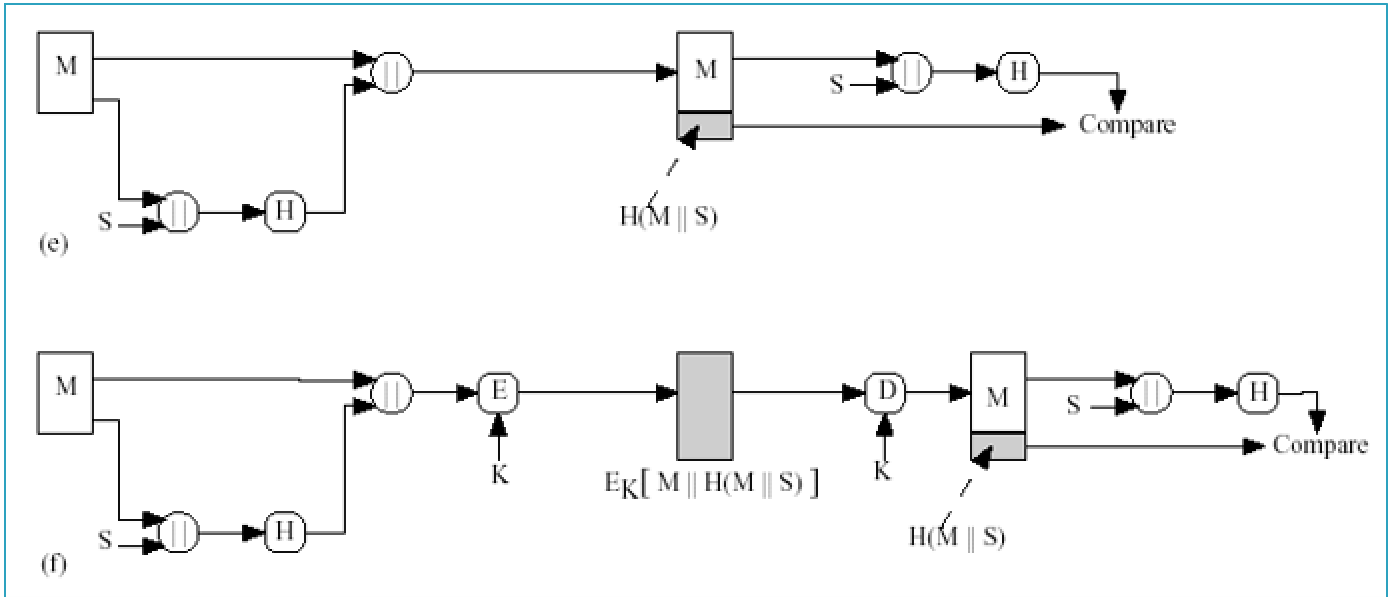
M:	Message
H:	Hash Function
E:	Block Cipher Encryption
D:	Block Cipher Decryption
¦¦:	Concatenation operation

Note:

- Scenario (a) (and (f) provide confidentiality and message authentication
- Scenario (b) (and (c)) provide message authentication only

Basic Uses of HMACs

- In scenarios (e) and (f) a symmetric secret seed S is used, which is shared between sender and receiver
- S is used to authenticate all messages exchanged between both endpoints
- Scenario (f) also uses a symmetric key K for confidentiality, which is independent from S



Case Study HMAC

- Assume you operate a distributed weather station with battery-operated sensors located across Ireland
- You use "public" networks (i.e. Wi-Fi, Internet) to collect data and send it for processing to a central hub in Galway
- Which basic uses of a Hash function as shown in the previous slides would be most **appropriate and efficient**?

Requirements for a Hash Function H(x)

- **One-way property** (also called **pre-image resistance**): For a given hash function H and a hash value h it is infeasible to find x such that $\mathrm{H(x)} = \mathrm{h}$
  - I.e., it is virtually impossible to generate a message given a hash
  - Such a situation is also called a **hash collision**
- Why is the one-way property important?
  - See Figure (e): An opponent could intercept M | | H(M, S), create inputs M | | X (with some random value X), until a hash collision is found (i.e. S)

Requirements for a Hash Function H(x)

- **Weak collision resistance** (also called **second pre-image resistance**):
  For a given hash function H and a known input x it is infeasible to find another input y with
  y != x and H(x) = H(y)

- Why is the weak collision resistance important?
  - See Figure (b): An opponent could
    - calculate h(M) (as both h and M are known)
    - find an alternate message with the same hash code (a hash collision), and
    - send it together with the encrypted (original) hash code to the receiver
  - The receiver would not be able to realise that the original message had been tampered with
    - Think of the previous software patch example

Requirements for a Hash Function H(x)

- **Strong collision resistance** (also called **collision resistance**): It is computational infeasible to find **any** pair of inputs (i.e., messages) (x, y) with $H(x) = H(y)$

- Why is the strong collision resistance important?
  - Again, see Figure (b), but this time the attack vector is different:
    - Rather than intercepting a hashed message in transit, the attacker presents the signing authority a crafted authentic message that has the same hash as a fraudulent message
    - Generating such a crafted message is accommodated by the **Birthday Paradox** discussed earlier

Birthday Paradox Attack

- Rather than thinking of birthdays, we consider messages and their hashes
- In the BPA the attacker does not intercept a hashed message in transit, but presents the signing authority a crafted authentic message that has the same hash as a fraudulent message (HMAC use case b)
- For a hash value that is m-bit long, the attacker creates a large number (i.e., in the order of $2^{0.5m}$) of variations of:
  - correct messages
  - fraudulent replacement messages
- The birthday paradox will make it more likely to find among both sets a correct message M_{nice} that has the same hash as a fraudulent message M_{nasty}
- M_{nice} is presented to the signing authority, who
  - hashes the message
  - encrypt the hash using the secret key (only known to the signing authority and the receiver)
  - concatenate message and hash
- Before the message is sent off, the attacker replaces M_{nice} with $_{Mnasty}$
- The receiver gets M_{nasty}, but will assume that it was signed (and send) by the signing authority

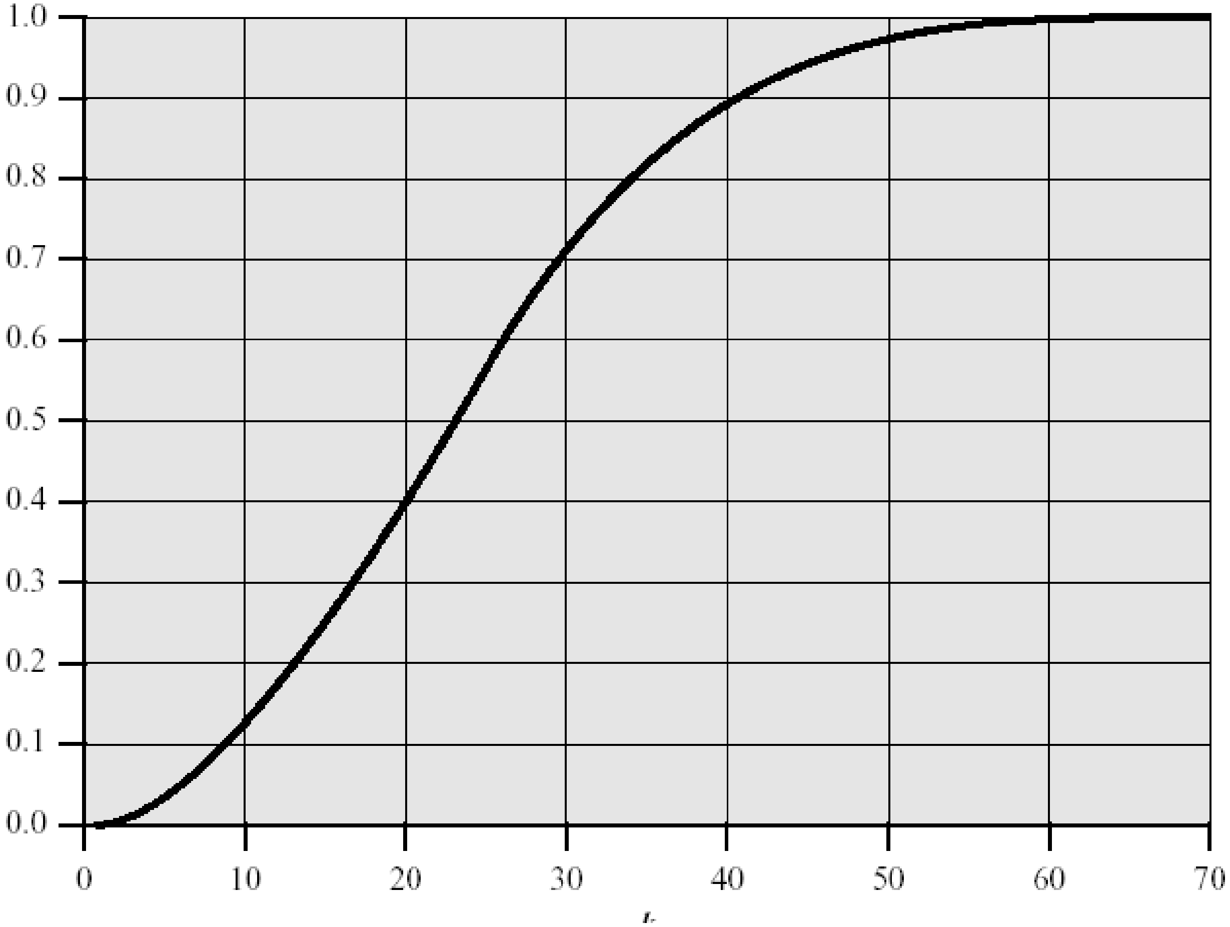
Birthday Paradox

- What is the minimum value k such that the probability is greater than 0.5 that at least 2 people in a group of k people have the same birthday, assuming that a year has 365 days?

- Intuitively someone would assume that $k = 365 / 2 = 183$

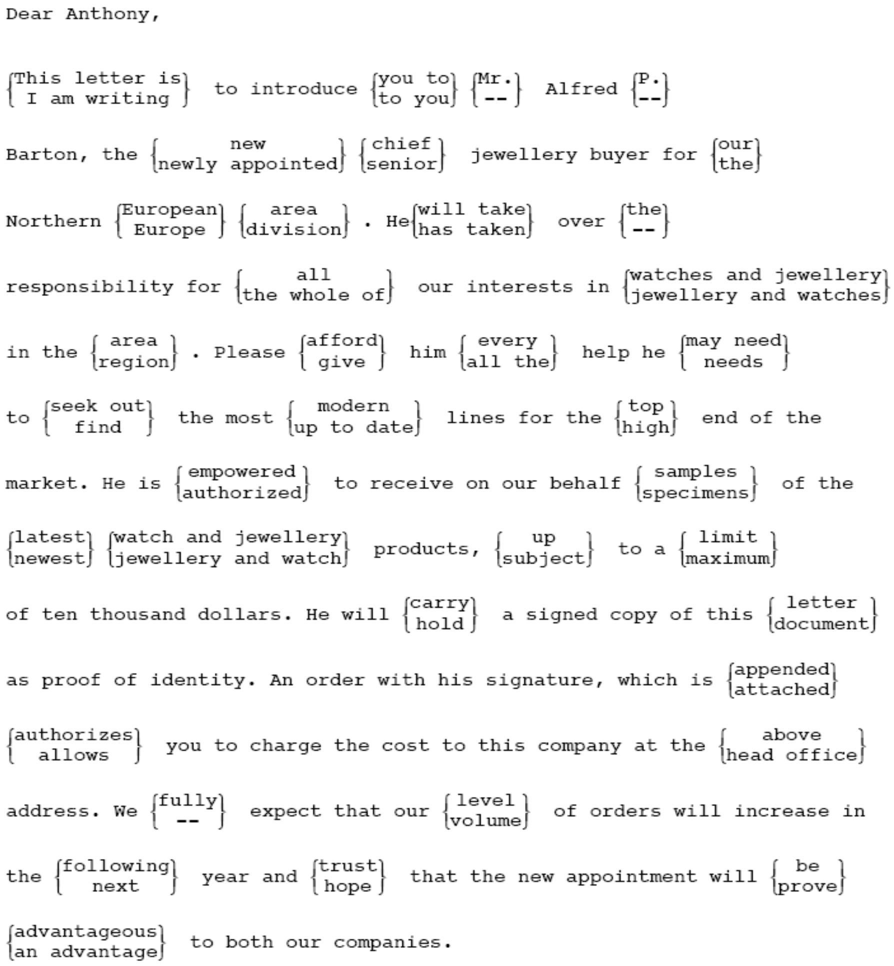
- **Probability theory shows, that $k = 23$ is sufficient!**

Birthday Paradox



BPA – How to create many Variations of a Message

- The example gives a letter in 2^{37} variations



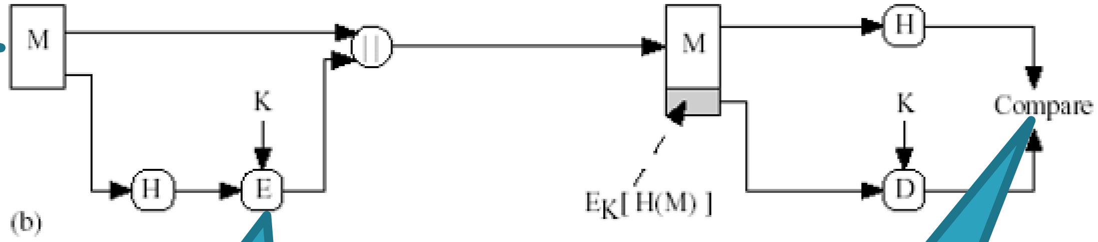
Dear Anthony,

$\begin{Bmatrix}\text{This letter is}\\\text{I am writing}\end{Bmatrix}$ to introduce $\begin{Bmatrix}\text{you to}\\\text{to you}\end{Bmatrix}$ $\begin{Bmatrix}\text{Mr.}\\\text{--}\end{Bmatrix}$ Alfred $\begin{Bmatrix}\text{P.}\\\text{--}\end{Bmatrix}$ Barton, the $\begin{Bmatrix}\text{new}\\\text{newly appointed}\end{Bmatrix}$ $\begin{Bmatrix}\text{chief}\\\text{senior}\end{Bmatrix}$ jewellery buyer for $\begin{Bmatrix}\text{our}\\\text{the}\end{Bmatrix}$ Northern $\begin{Bmatrix}\text{European}\\\text{Europe}\end{Bmatrix}$ $\begin{Bmatrix}\text{area}\\\text{division}\end{Bmatrix}$. He $\begin{Bmatrix}\text{will take}\\\text{has taken}\end{Bmatrix}$ over $\begin{Bmatrix}\text{the}\\\text{--}\end{Bmatrix}$ responsibility for $\begin{Bmatrix}\text{all}\\\text{the whole of}\end{Bmatrix}$ our interests in $\begin{Bmatrix}\text{watches and jewellery}\\\text{jewellery and watches}\end{Bmatrix}$ in the $\begin{Bmatrix}\text{area}\\\text{region}\end{Bmatrix}$. Please $\begin{Bmatrix}\text{afford}\\\text{give}\end{Bmatrix}$ him $\begin{Bmatrix}\text{every}\\\text{all the}\end{Bmatrix}$ help he $\begin{Bmatrix}\text{may need}\\\text{needs}\end{Bmatrix}$ to $\begin{Bmatrix}\text{seek out}\\\text{find}\end{Bmatrix}$ the most $\begin{Bmatrix}\text{modern}\\\text{up to date}\end{Bmatrix}$ lines for the $\begin{Bmatrix}\text{top}\\\text{high}\end{Bmatrix}$ end of the market. He is $\begin{Bmatrix}\text{empowered}\\\text{authorized}\end{Bmatrix}$ to receive on our behalf $\begin{Bmatrix}\text{samples}\\\text{specimens}\end{Bmatrix}$ of the $\begin{Bmatrix}\text{latest}\\\text{newest}\end{Bmatrix}$ $\begin{Bmatrix}\text{watch and jewellery}\\\text{jewellery and watch}\end{Bmatrix}$ products, $\begin{Bmatrix}\text{up}\\\text{subject}\end{Bmatrix}$ to a $\begin{Bmatrix}\text{limit}\\\text{maximum}\end{Bmatrix}$ of ten thousand dollars. He will $\begin{Bmatrix}\text{carry}\\\text{hold}\end{Bmatrix}$ a signed copy of this $\begin{Bmatrix}\text{letter}\\\text{document}\end{Bmatrix}$ as proof of identity. An order with his signature, which is $\begin{Bmatrix}\text{appended}\\\text{attached}\end{Bmatrix}$ $\begin{Bmatrix}\text{authorizes}\\\text{allows}\end{Bmatrix}$ you to charge the cost to this company at the $\begin{Bmatrix}\text{above}\\\text{head office}\end{Bmatrix}$ address. We $\begin{Bmatrix}\text{fully}\\\text{--}\end{Bmatrix}$ expect that our $\begin{Bmatrix}\text{level}\\\text{volume}\end{Bmatrix}$ of orders will increase in the $\begin{Bmatrix}\text{following}\\\text{next}\end{Bmatrix}$ year and $\begin{Bmatrix}\text{trust}\\\text{hope}\end{Bmatrix}$ that the new appointment will $\begin{Bmatrix}\text{be}\\\text{prove}\end{Bmatrix}$ $\begin{Bmatrix}\text{advantageous}\\\text{an advantage}\end{Bmatrix}$ to both our companies.

Case Study: Circulating Software using the BPA

- This is a typical insider attack (here conducted by Grumpy George – GG – a disgruntled lead engineer in your team)
- Again, your team develops an urgent software patch, which is hashed
- The 32-bit hash value is encoded using a symmetric key K, which is shared with your client
- The key is only known to you and you client, but not to GG

Software patch



Your authenticator (encrypted hash)

Client validates software patch

Case Study: Circulating Software via a Birthday Paradox Attack

- GG as the lead engineer creates a large number of binary code versions for
  - software patches (to be presented to quality team)
  - malicious software patches (to be circulated)
- How can GG create > 2 * 2^{16} different source code variations?
  - GG introduces in both source code files a new constant variable (e.g. long int) that is not otherwise used, e.g.
    …
    const unsigned long int var = 12; // possible values are 0 … 2^{64}-1
  - GG then creates different source codes by systematically incrementing var
    - GG is able to create 2^{64} different versions of both programs if needs to be
- GG compiles each of those software versions and calculates their hash
- GG looks for a hash collision, i.e. a software patch and a malicious patch that have the same hash code
- GG present this software patch to quality team, who sign it using key K
- GG replaces the software with the malicious patch before sending it to the client

Hash Function Execution (Example HAVAL)

- HAVAL creates a 256-bit fingerprint, for example:
  - "The quick brown fox jumps over the lazy **d**og“
    will be translated into the (256 bit) hash
    “b89c551cdfe2e06dbd4cea2be1bc7d557416c58ebb4d07cbc94e49f710c55be4”
  - “The quick brown fox jumps over the lazy **c**og”
    will be translated into the hash
    “60983bb8c8f49ad3bea29899b78cd741f4c96e911bbc272e5550a4f195a4077e”
- **I.e. very similar inputs result in totally different outputs, there is no correlation between a hash and its original input**

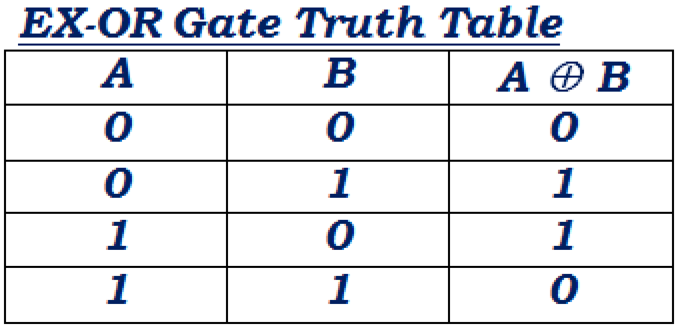
A naive Hash Function based on XOR

- Consider the XOR function ⊕:
- The input is broken into m blocks
- For the resulting hash value C, each bit C_i is calculated via

$$\mathbf{C_i = b_{i1} \oplus b_{i2} \oplus b_{i3} \oplus \dots\ b_{im}}$$

Where

- m = the number of n-bit blocks and
- b_{ij} is the i^{th} bit of the j^{th} block



EX-OR Gate Truth Table

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

A naive Hash Function based on XOR

	Bit 1	**Bit 2**	**…**	**Bit n**
Block 1	b_{11}	b_{21}		b_{n1}
Block 2	b_{12}	b_{22}		b_{n2}
…				
Block m	b_{1m}	b_{2m}		b_{nm}
Hash code	C_1	C_2		C_n

A naive Hash Function based on XOR

- Consider the ASCII-encoded input “ABC” and a hash function H that calculates an 8-bit hash h:
  - ASCII(A) = 65_{10} = 01000001_{2}
  - ASCII(B) = 66_{10} = 01000010_{2}
  - ASCII(C) = 67_{10} = 01000011_{2}

	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
A	0	1	0	0	0	0	0	1
B	0	1	0	0	0	0	1	0
C	0	1	0	0	0	0	1	1
h	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**

H(“ABC”) = h = 64_{10} = “@”

A naive Hash Function based on XOR

- Does this algorithm fulfil the requirements of a hash function:
  - One-way property?
  - Weak collision resistance?

	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
A	0	1	0	0	0	0	0	1
B	0	1	0	0	0	0	1	0
C	0	1	0	0	0	0	1	1
h	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**

H(“ABC”) = 64_{10} = “@”

Example: 8-bit Hash Function based on XOR

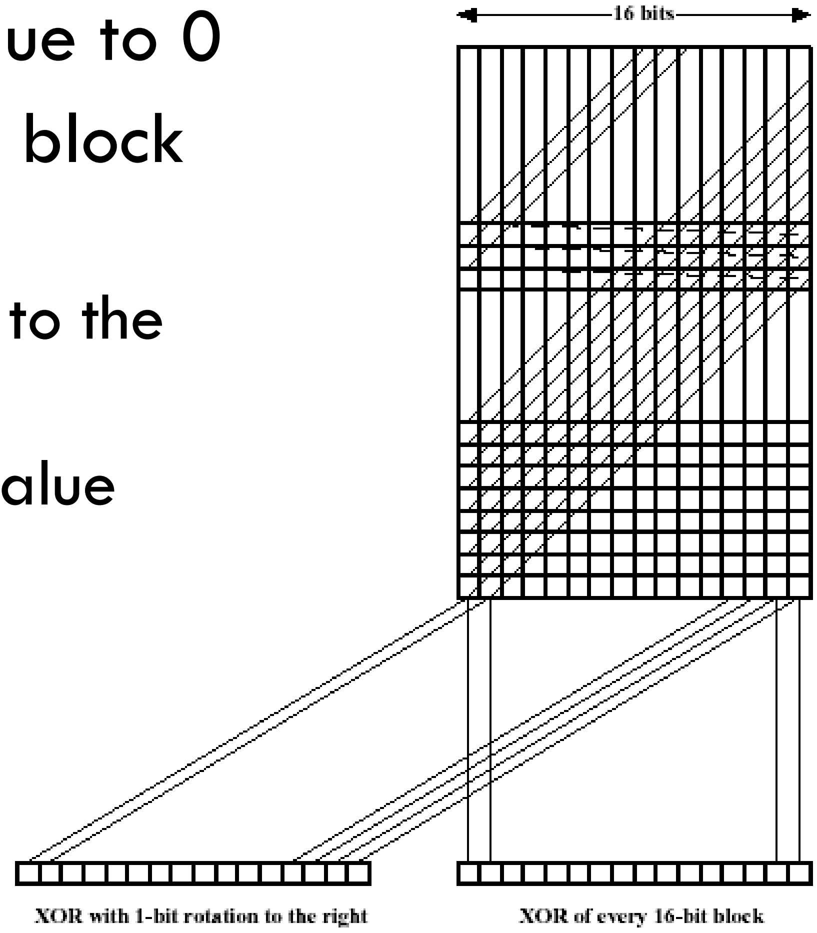
- Fulfils requirements of hash function?
  - One-way property? Certainly not!
  - Weak collision resistance? H(“ABC”) = H(“@@@”) = H(“@@@@@@“) = …

	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
“@“	0	1	0	0	0	0	0	0
“@“	0	1	0	0	0	0	0	0
“@“	0	1	0	0	0	0	0	0
h	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**

H(“@@@”) = 64_{10} = “@”

A naive Hash Function based on rotating XOR

- Initially set the n-bit hash value to 0
- Process each successive n-bit block a follows:
  - Rotate the current hash value to the left by one bit
  - XOR the block into the hash value


16 bits
XOR with 1-bit rotation to the right
XOR of every 16-bit block

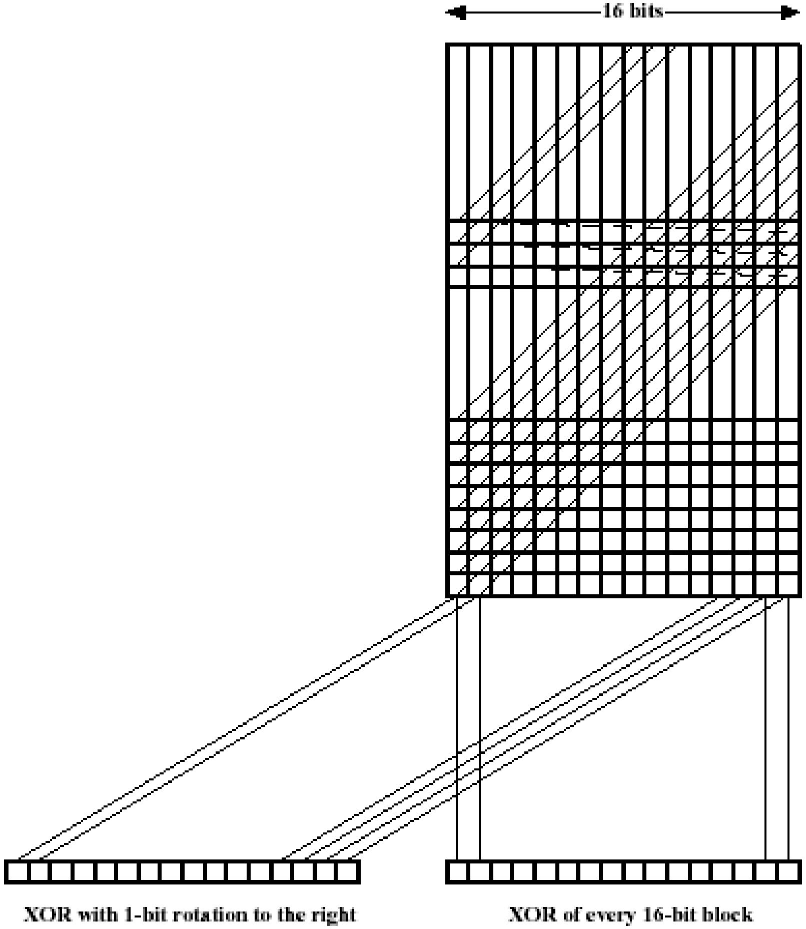
Example: Simple Hash Function based on Rotating XOR

- Consider “ABCD”
- “AB” = $01000010\ 01000011_2$
- “CD” = $01000100\ 01000101_2$
- “CD” left-rotated = $10001000\ 10001010_2$

	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
	0	1	0	0	0	0	1	1	0	1	0	0	0	0	1	1
	1	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0
h	1	1	0	0	1	0	1	1	1	1	0	0	1	0	0	1

h = $CBC9_{16}$

Example: Simple Hash Function based on Rotating XOR

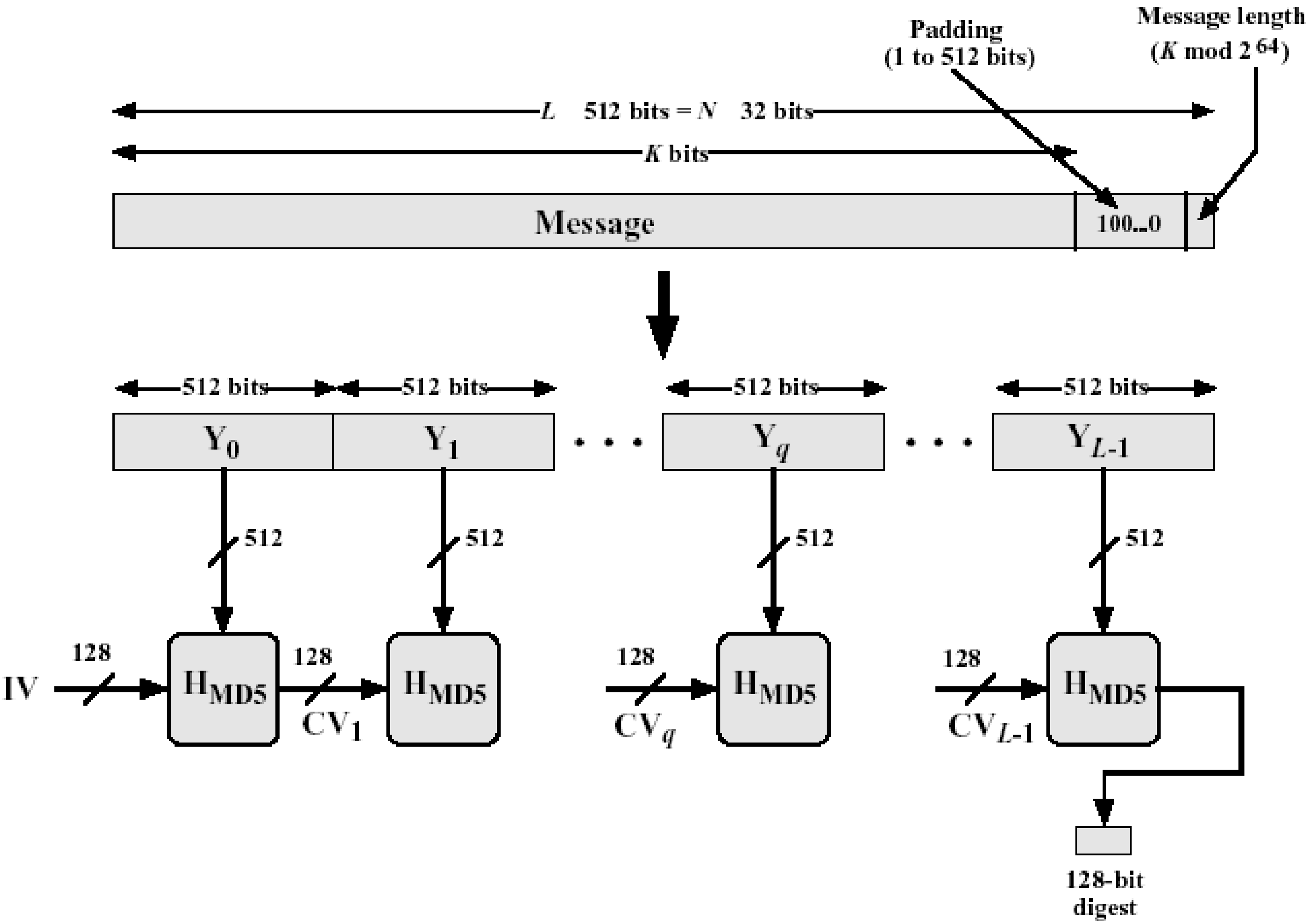

16 bits
XOR with 1-bit rotation to the right
XOR of every 16-bit block

- Assume a password must be at least 2 ASCII-encoded characters long
- Fulfils requirements of hash Function?
  - One-way property?
  - Weak collision resistance?

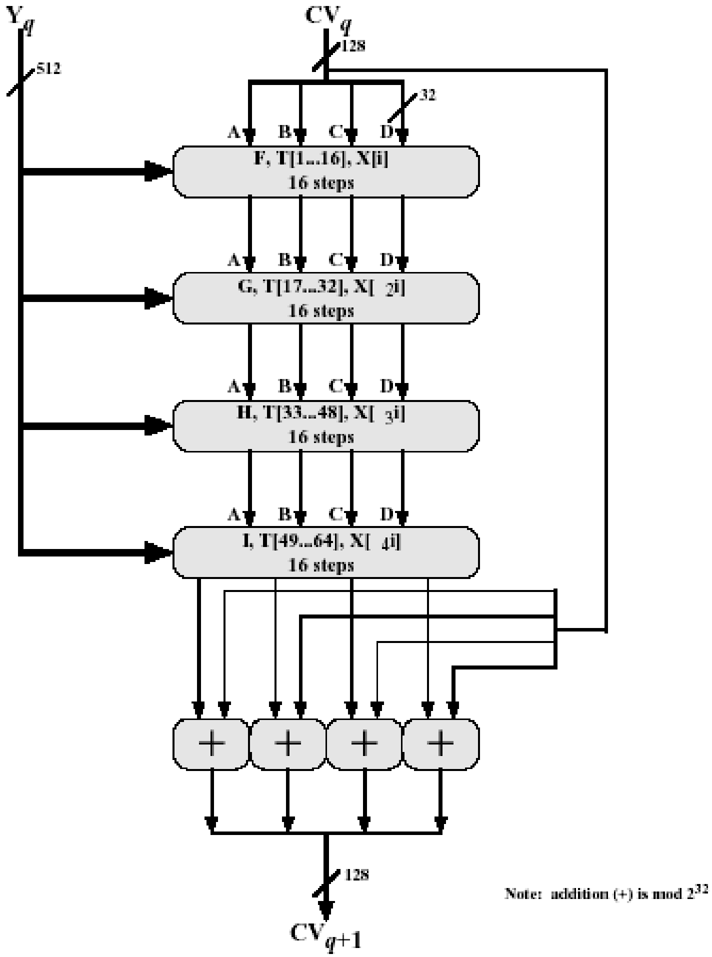
Examples for Hash Algorithms

- In order to meet the aforementioned requirements, a hash algorithm must
  - be non-trivial
  - calculate long hash values
- Popular hash functions include:
  - MD5:
    - Produces a 128-bit hash value
    - Specified as Internet standards (RFC1321)
    - Still has some popularity, but unsafe for years (broken via collision attacks)
  - SHA (Secure Hash Algorithm) - X:
    - Family of hash functions, designed by NIST & NSA
    - SHA-3 (released 2015) produces 224-, 256-, 384- and 512-bits hash values
    - Internet standard
  - RIPEMD-160:
    - Creates a 160-bit hash value
    - Developed in Europe
- See https://defuse.ca/checksums.htm

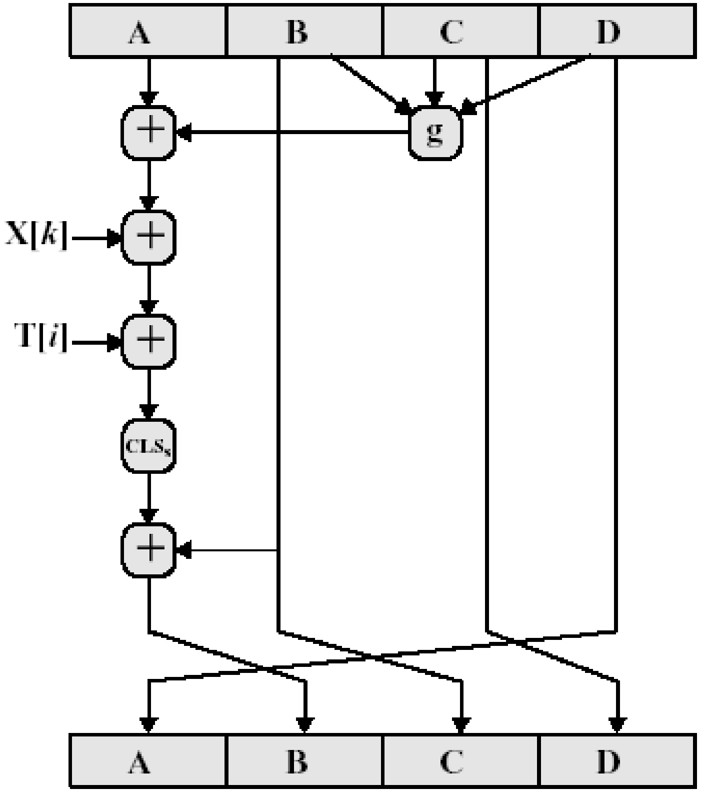
FYI: MD5-An Overview



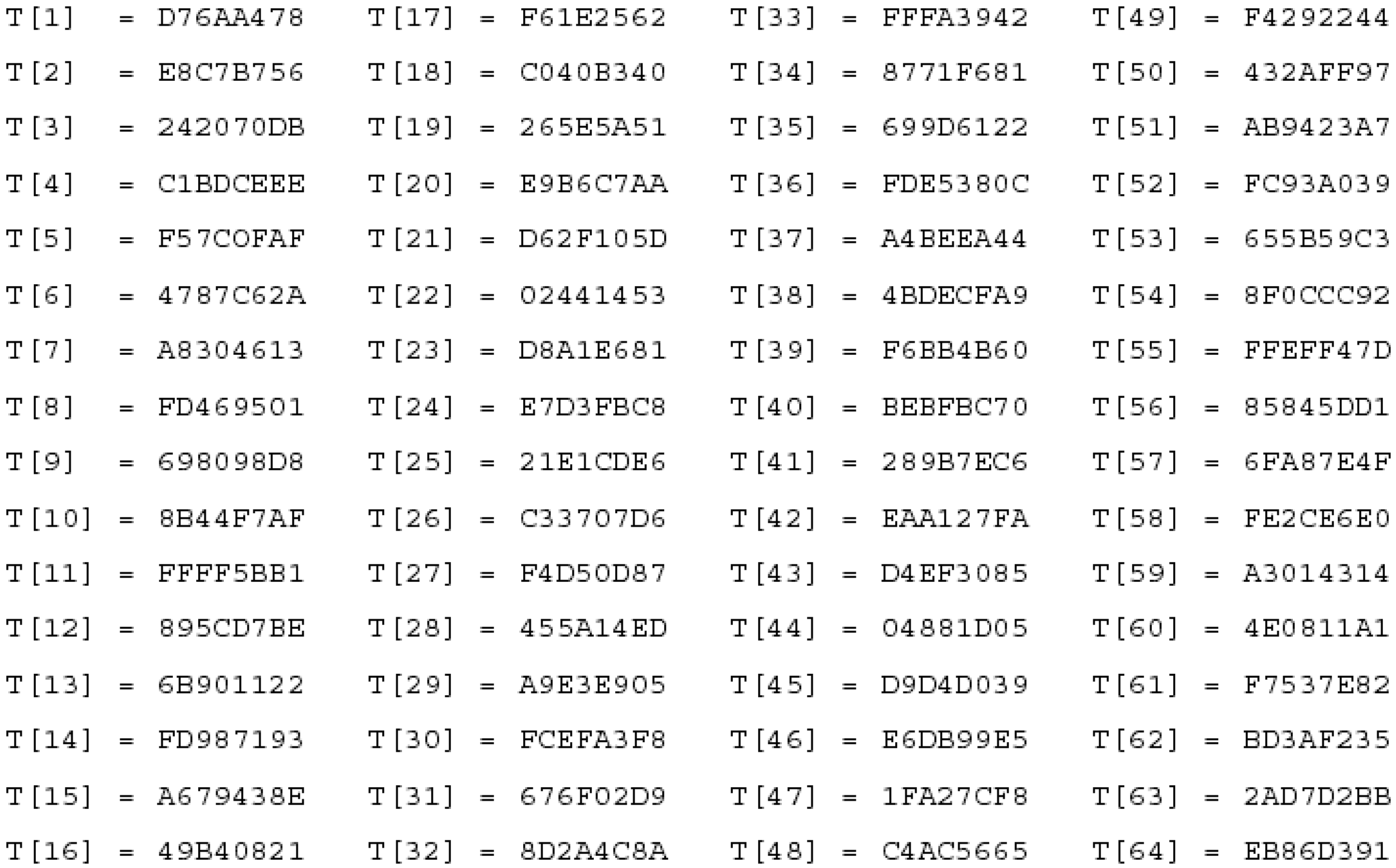
FYI: MD5-Processing of a Single 512 Bit Block (left) and Elementary MD5 Operation



Note: addition (+) is mod 2^{32}



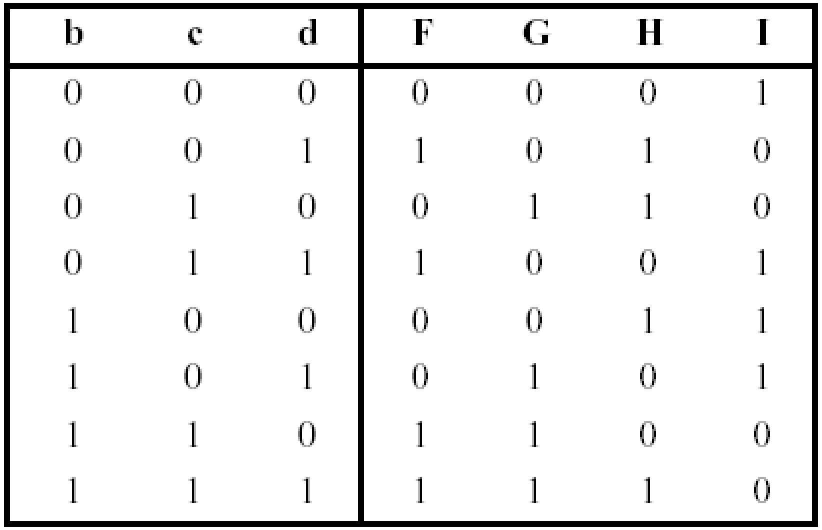
FYI: MD5-Table T



T[1]	=	D76AA478	T[17]	=	F61E2562	T[33]	=	FFFA3942	T[49]	=	F4292244
T[2]	=	E8C7B756	T[18]	=	C040B340	T[34]	=	8771F681	T[50]	=	432AFF97
T[3]	=	242070DB	T[19]	=	265E5A51	T[35]	=	699D6122	T[51]	=	AB9423A7
T[4]	=	C1BDCEEE	T[20]	=	E9B6C7AA	T[36]	=	FDE5380C	T[52]	=	FC93A039
T[5]	=	F57COFAF	T[21]	=	D62F105D	T[37]	=	A4BEEA44	T[53]	=	655B59C3
T[6]	=	4787C62A	T[22]	=	02441453	T[38]	=	4BDECFA9	T[54]	=	8F0CCC92
T[7]	=	A8304613	T[23]	=	D8A1E681	T[39]	=	F6BB4B60	T[55]	=	FFEFF47D
T[8]	=	FD469501	T[24]	=	E7D3FBC8	T[40]	=	BEBFBC70	T[56]	=	85845DD1
T[9]	=	698098D8	T[25]	=	21E1CDE6	T[41]	=	289B7EC6	T[57]	=	6FA87E4F
T[10]	=	8B44F7AF	T[26]	=	C33707D6	T[42]	=	EAA127FA	T[58]	=	FE2CE6E0
T[11]	=	FFFF5BB1	T[27]	=	F4D50D87	T[43]	=	D4EF3085	T[59]	=	A3014314
T[12]	=	895CD7BE	T[28]	=	455A14ED	T[44]	=	04881D05	T[60]	=	4E0811A1
T[13]	=	6B901122	T[29]	=	A9E3E905	T[45]	=	D9D4D039	T[61]	=	F7537E82
T[14]	=	FD987193	T[30]	=	FCEFA3F8	T[46]	=	E6DB99E5	T[62]	=	BD3AF235
T[15]	=	A679438E	T[31]	=	676F02D9	T[47]	=	1FA27CF8	T[63]	=	2AD7D2BB
T[16]	=	49B40821	T[32]	=	8D2A4C8A	T[48]	=	C4AC5665	T[64]	=	EB86D391

FYI: MD5-Primitive Functions and their Truth Tables

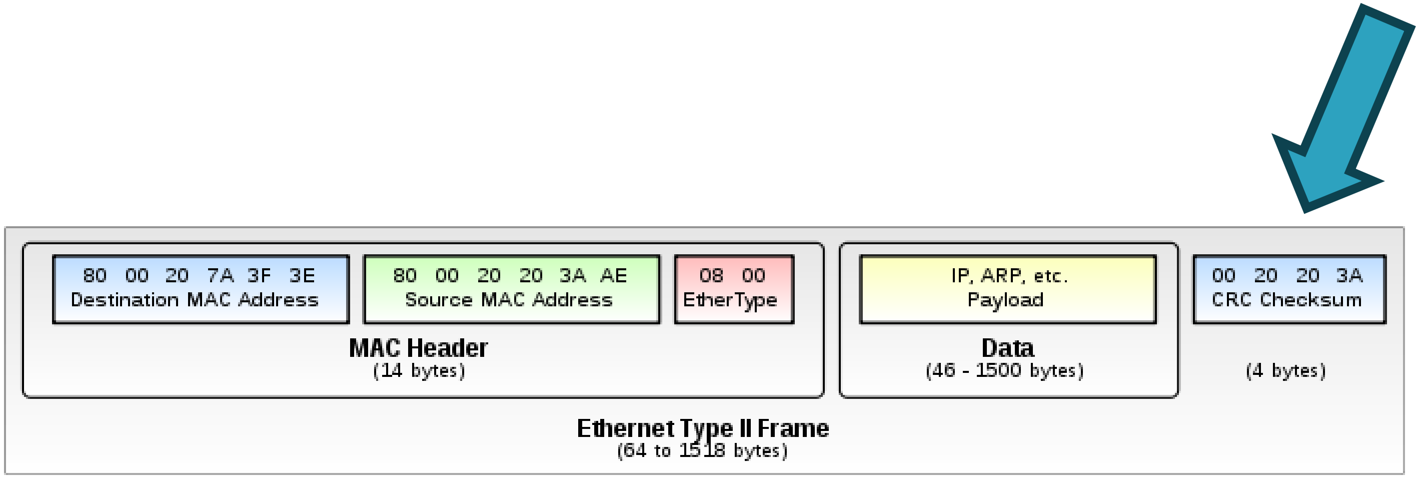
Round	Primitive function g	g(b, c, d)
1	F(b, c, d)	(b AND c) OR (NOT b AND d)
2	G(b, c, d)	(b AND d) OR (c AND NOT d)
3	H(b, c, d)	B EXOR c EXOR d
4	I(a, b, c)	C EXOR (b or NOT d)



b	c	d	F	G	H	I
0	0	0	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	0	1	1
1	0	1	0	1	0	1
1	1	0	1	1	0	0
1	1	1	1	1	1	0

Non-Cryptographic Hash Functions aka Checksums

- Checksums are designed to detect bit errors of files or data streams, e.g.
  - Hard disk storage errors
  - Data transmission errors
- CRC (Cyclic Redundancy Code) is a well know example
- Such checksums are too short and vulnerable to brute force attacks, and **are not suitable for cryptographic purposes**



CT437
COMPUTER SECURITY AND FORENSIC COMPUTING

HASH CRACKING AND RAINBOW TABLES

Dr. Michael Schukat

OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Lecture Overview

- Methods to reverse-engineer hashed passwords
- Rainbow tables
- A recap on SQL injection attacks (based on CT417 content), i.e.
  - SQL
  - HTTP get / post Methods and PHP
  - SQL injection attacks
  - SQL injection attack mitigation strategies

can be found at the end of this slide deck

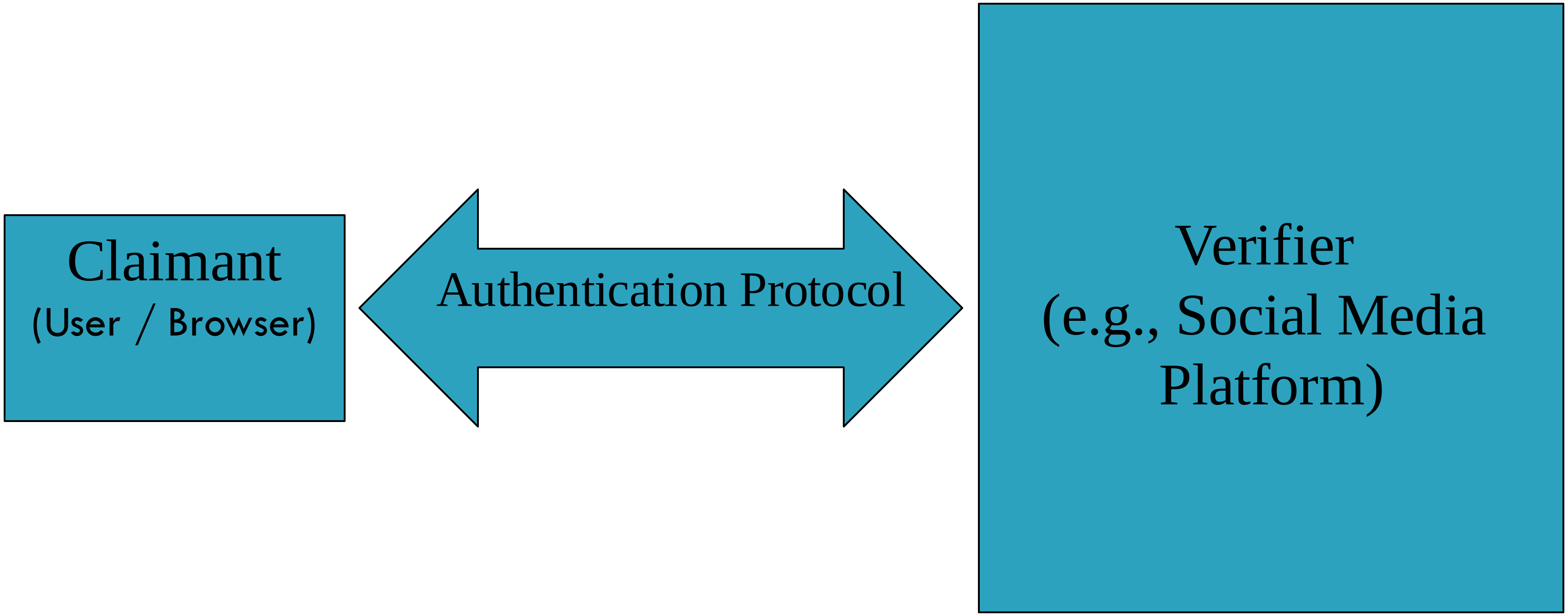
Lecture Motivation

- One-way property, weak and strong collision resistance are fundamental properties of a hash function
- These come also into play when we consider common password storage methods …
- … and approaches to undermine such methods
- Such approaches are summarised in this slide deck

What is a Password?

- A memorised secret used to confirm the identity of a user
  - Typically, an arbitrary string of characters including letters, digits, or other symbols
  - A purely numeric secret is called a personal identification number (PIN)
- The secret is memorised by a party called the **claimant** while the party verifying the identity of the claimant is called the **verifier**
- Claimant and verifier communicate via an **authentication protocol**

Claimant and Verifier



Claimant
(User / Browser)

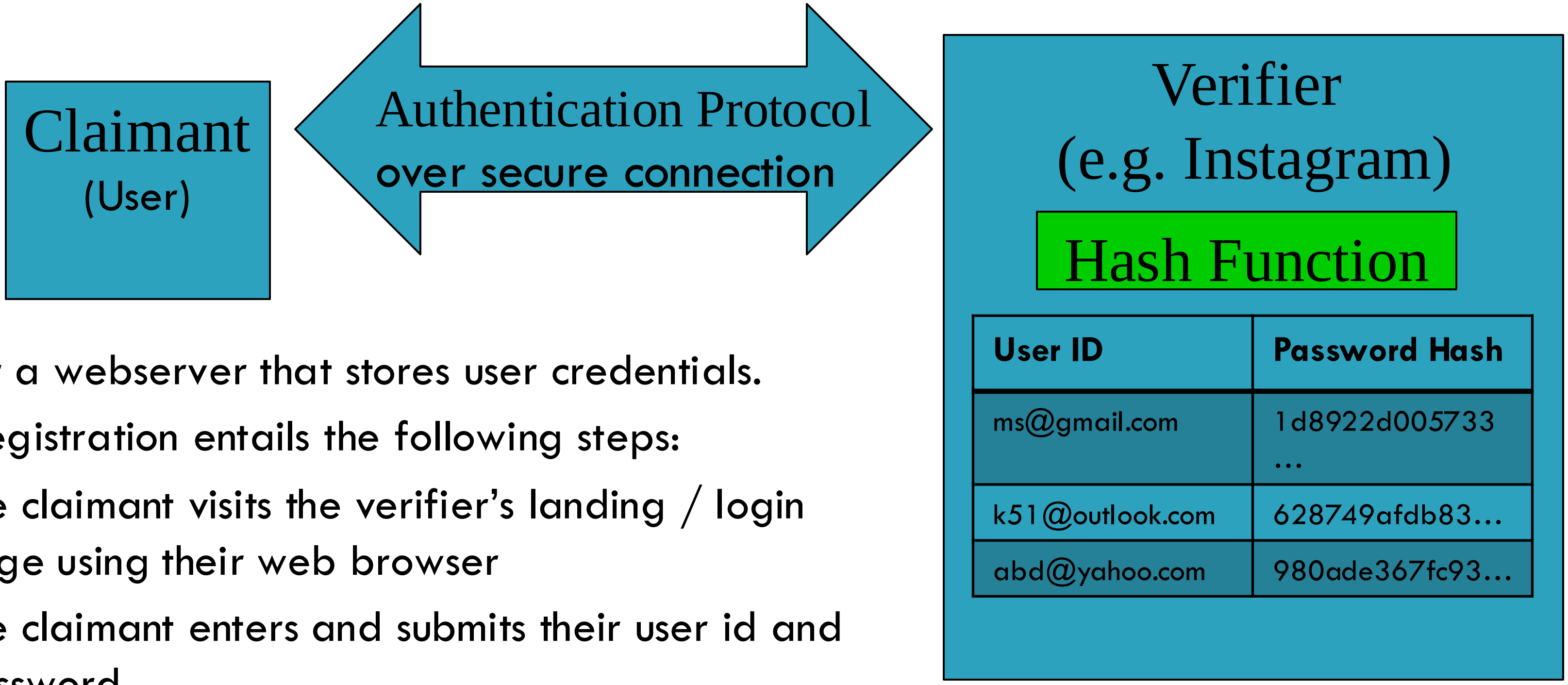
Authentication Protocol

Verifier
(e.g., Social Media Platform)

Storing User Passwords

- User passwords at rest (e.g., in database tables) are hashed instead of being stored in plaintext
- Idea:
  - "KenSentMe!" → "7b24afc8bc80e548d66c4e7ff72171c5"
    - Note: This token is in hex format, it is128 bit long (32 x 4 bits)
  - An attacker cannot algorithmically reverse-engineer a hash function to recover the original password
    - Recall hash function properties
  - The verifier does not have a plaintext copy of the password either

Why is this Form of Password Hash Management problematic?



Claimant
(User)

Authentication Protocol
over secure connection

Verifier
(e.g. Instagram)

Hash Function

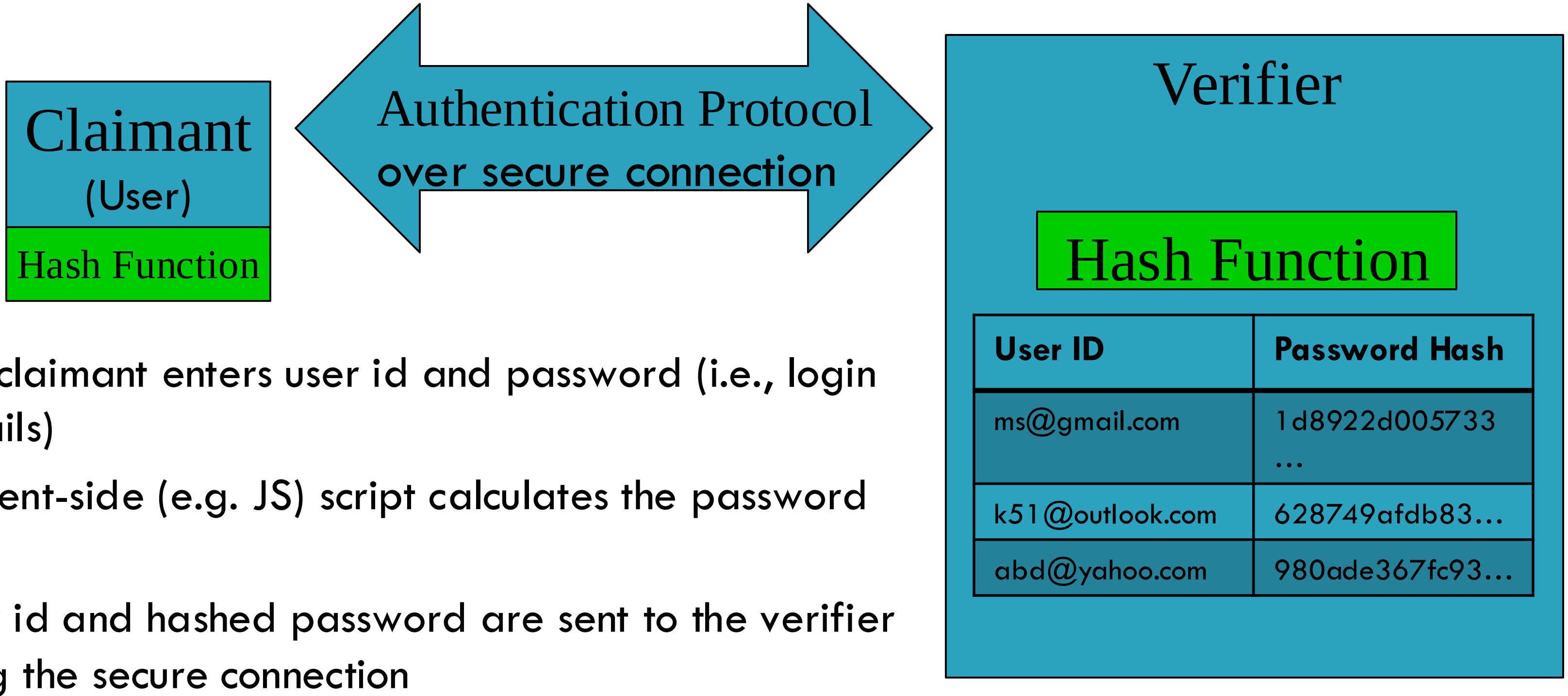
User ID	Password Hash
ms@gmail.com	1d8922d005733 …
k51@outlook.com	628749afdb83…
abd@yahoo.com	980ade367fc93…

Consider a webserver that stores user credentials.

A user registration entails the following steps:

1. The claimant visits the verifier's landing / login page using their web browser
2. The claimant enters and submits their user id and password
3. Both are sent to the verifier over the secure connection
4. The verifier calculates the hash, and and stores it together with the user name in the DB table

Server-Side Password Storage



User ID	Password Hash
ms@gmail.com	1d8922d005733 …
k51@outlook.com	628749afdb83…
abd@yahoo.com	980ade367fc93…

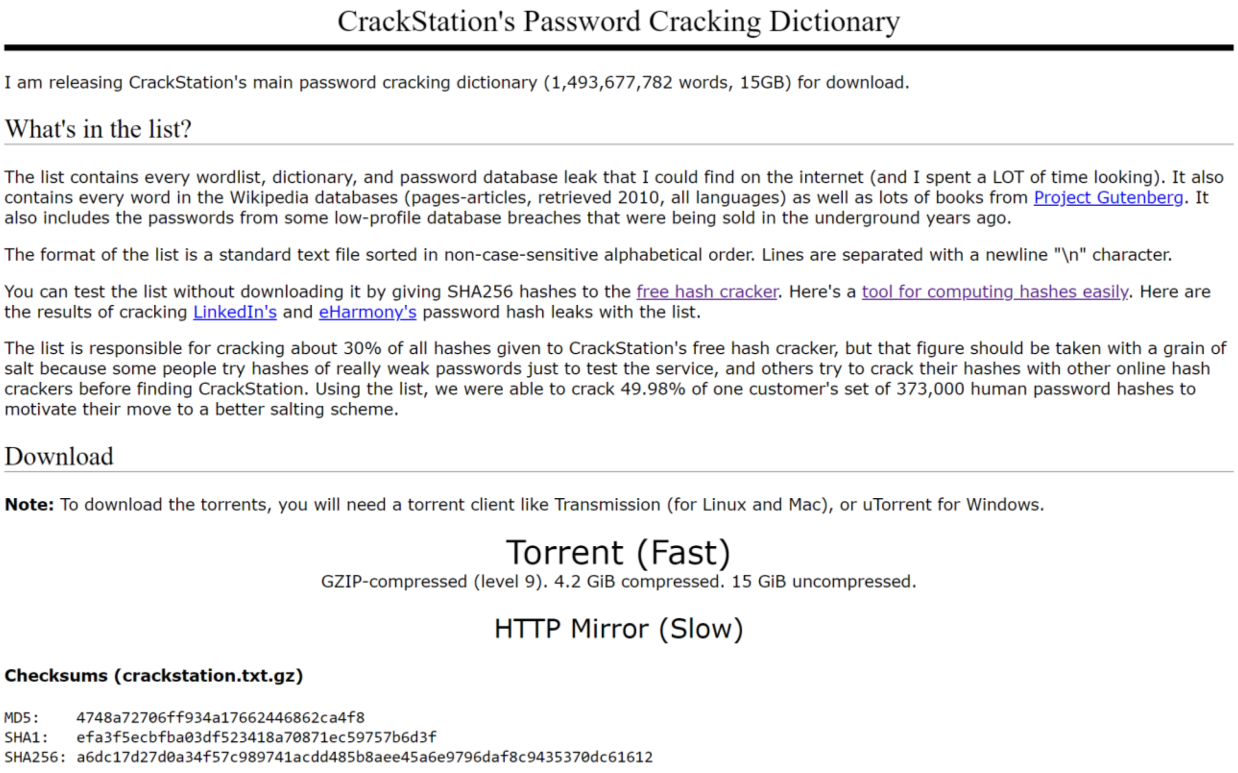
1. The claimant enters user id and password (i.e., login details)
2. A client-side (e.g. JS) script calculates the password hash
3. User id and hashed password are sent to the verifier using the secure connection
4. The verifier checks if the transmitted user id and hashed password against the stored values in the table
5. The verifier notifies the claimant via the authentication protocol if the authentication was successful

Dictionary-Based Brute-Force Search

- Assume an attacker retrieves an entire DB table containing user IDs and hashed passwords
- Hash functions are one-way functions, so hash values cannot be transformed back to the original input
- However, assuming that a user picks a common word or phrase, or a known password as their own password, a simple dictionary search can be used to systematically identify a match for a given hash value
  - Here the underlying hash function must be known
- Such dictionaries are based on large word, phrase or password collections
- ☺ :
  - Straight forward process
  - Large dictionaries are readily available (next slide)
- ☹ :
  - Significant computational effort to find match
  - No guaranteed result

CrackStation's Password Cracking Dictionary

- https://crackstation.net/crackstation-wordlist-password-cracking-dictionary.htm



CrackStation's Password Cracking Dictionary

I am releasing CrackStation's main password cracking dictionary (1,493,677,782 words, 15GB) for download.

What's in the list?

The list contains every wordlist, dictionary, and password database leak that I could find on the internet (and I spent a LOT of time looking). It also contains every word in the Wikipedia databases (pages-articles, retrieved 2010, all languages) as well as lots of books from Project Gutenberg. It also includes the passwords from some low-profile database breaches that were being sold in the underground years ago.

The format of the list is a standard text file sorted in non-case-sensitive alphabetical order. Lines are separated with a newline "\n" character.

You can test the list without downloading it by giving SHA256 hashes to the free hash cracker. Here's a tool for computing hashes easily. Here are the results of cracking LinkedIn's and eHarmony's password hash leaks with the list.

The list is responsible for cracking about 30% of all hashes given to CrackStation's free hash cracker, but that figure should be taken with a grain of salt because some people try hashes of really weak passwords just to test the service, and others try to crack their hashes with other online hash crackers before finding CrackStation. Using the list, we were able to crack 49.98% of one customer's set of 373,000 human password hashes to motivate their move to a better salting scheme.

Download

Note: To download the torrents, you will need a torrent client like Transmission (for Linux and Mac), or uTorrent for Windows.

Torrent (Fast)

GZIP-compressed (level 9). 4.2 GiB compressed. 15 GiB uncompressed.

HTTP Mirror (Slow)

Checksums (crackstation.txt.gz)

```
MD5:    4748a72706ff934a17662446862ca4f8
SHA1:   efa3f5ecbfba03df523418a70871ec59757b6d3f
SHA256: a6dc17d27d0a34f57c989741acdd485b8aee45a6e9796daf8c9435370dc61612
```

Example

- Assume a hash code and the underlying hash function are known
- The dictionary contains 10^{10} entries
- A single laptop / PC can compute 10^{5} hash values per second
- It takes 10^{5} seconds (~29 hours) to search the entire dictionary for a match
- This process can be vastly improved by using pre-processed lookup tables

Lookup Table-Based Attacks

- For a given hash function and dictionary
  - Calculate the hash values for all dictionary entries
  - Insert both values to a table (i.e. one line per entry)
  - Sort table (e.g. in ascending order of hash values)
    - Also called **lookup table**
  - Store the table
- Example table (assuming 44-bit hash values):

Hash value	**Password**
0x00000000354	gangster
0x00000001003	Bluemoon
0x00000001032	Z0om!
…	…

Lookup Table-Based Attacks

- A matching password for a given hash value can be recovered by systematically searching the look-up table via a binary search
- ☺ :
  - Such a table can be generated offline
  - The search process itself is fast (~$\log_2$(# of entries)) using binary search
    - A table containing 1.8x10^{19} entry would require just 64 guesses to find (or not) the correct password for a given hash value
- ☹ :
  - Huge table, with no guaranteed result
  - Different table required for every hash function

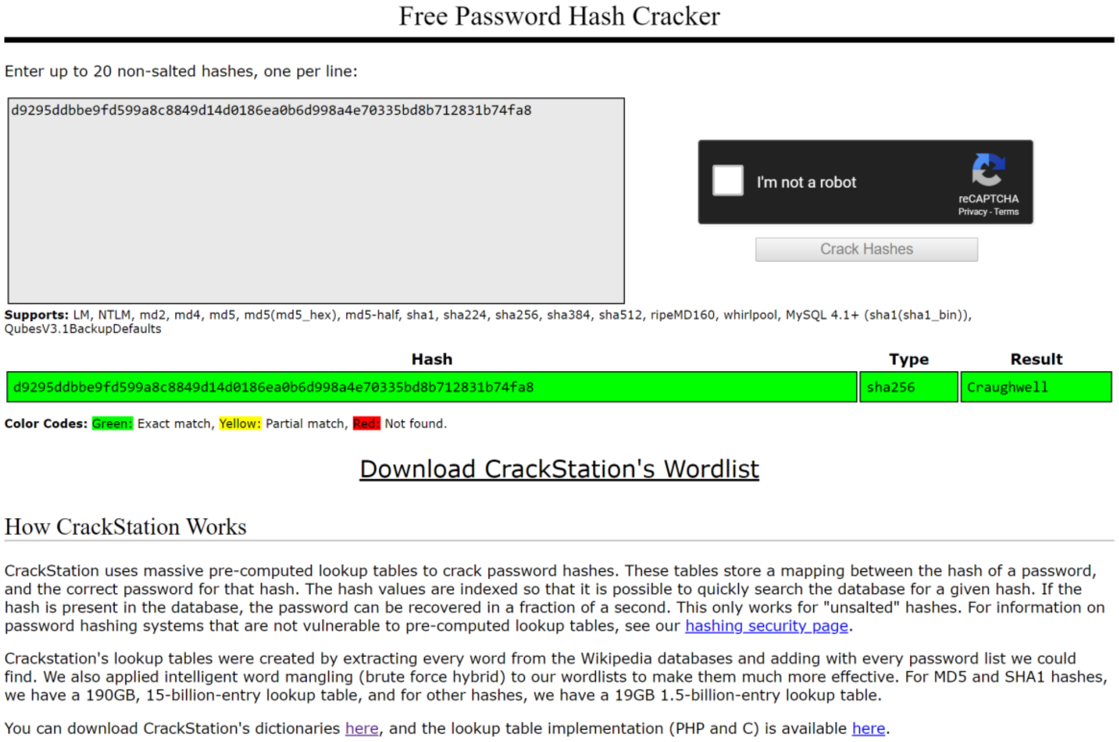
Lookup Table-Based Attacks: Example

- Assume a hash function that generates 16-byte (128 bit) hash values
- We calculate a lookup table for all possible 6-character long passwords composed of 64 possible characters A-Z, a-z, 0-9, "." and "/"
- A table would consist of 64^6 (= 68,719,476,736) entries, with every entry consisting of a 6-byte password and a 16 bytes hash
- **Total size of table ~ 1.4 Terabyte**

- However, there are online services available that host pre-computed look-up tables for password attacks (see next slide)

Crackstation’s free Password Hash Cracker

- □ https://crackstation.net/



Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
d9295ddbbe9fd599a8c8849d14d0186ea0b6d998a4e70335bd8b712831b74fa8
```

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
d9295ddbbe9fd599a8c8849d14d0186ea0b6d998a4e70335bd8b712831b74fa8	sha256	Craughwell

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Download CrackStation's Wordlist

How CrackStation Works

CrackStation uses massive pre-computed lookup tables to crack password hashes. These tables store a mapping between the hash of a password, and the correct password for that hash. The hash values are indexed so that it is possible to quickly search the database for a given hash. If the hash is present in the database, the password can be recovered in a fraction of a second. This only works for "unsalted" hashes. For information on password hashing systems that are not vulnerable to pre-computed lookup tables, see our hashing security page.

Crackstation's lookup tables were created by extracting every word from the Wikipedia databases and adding with every password list we could find. We also applied intelligent word mangling (brute force hybrid) to our wordlists to make them much more effective. For MD5 and SHA1 hashes, we have a 190GB, 15-billion-entry lookup table, and for other hashes, we have a 19GB 1.5-billion-entry lookup table.

You can download CrackStation's dictionaries here, and the lookup table implementation (PHP and C) is available here.

In-Class Activity: Password Recovery

- 5 minutes only, work alone or in a group
- What to do:
  - Pick a password and calculate its MD5 or SHA1 hash using https://defuse.ca/checksums.htm
  - Copy and paste the hash value into https://crackstation.net/ to see if it is can be recovered
  - Repeat the above and keep a list of all passwords
    - that **can** be cracked
    - that **cannot** be cracked

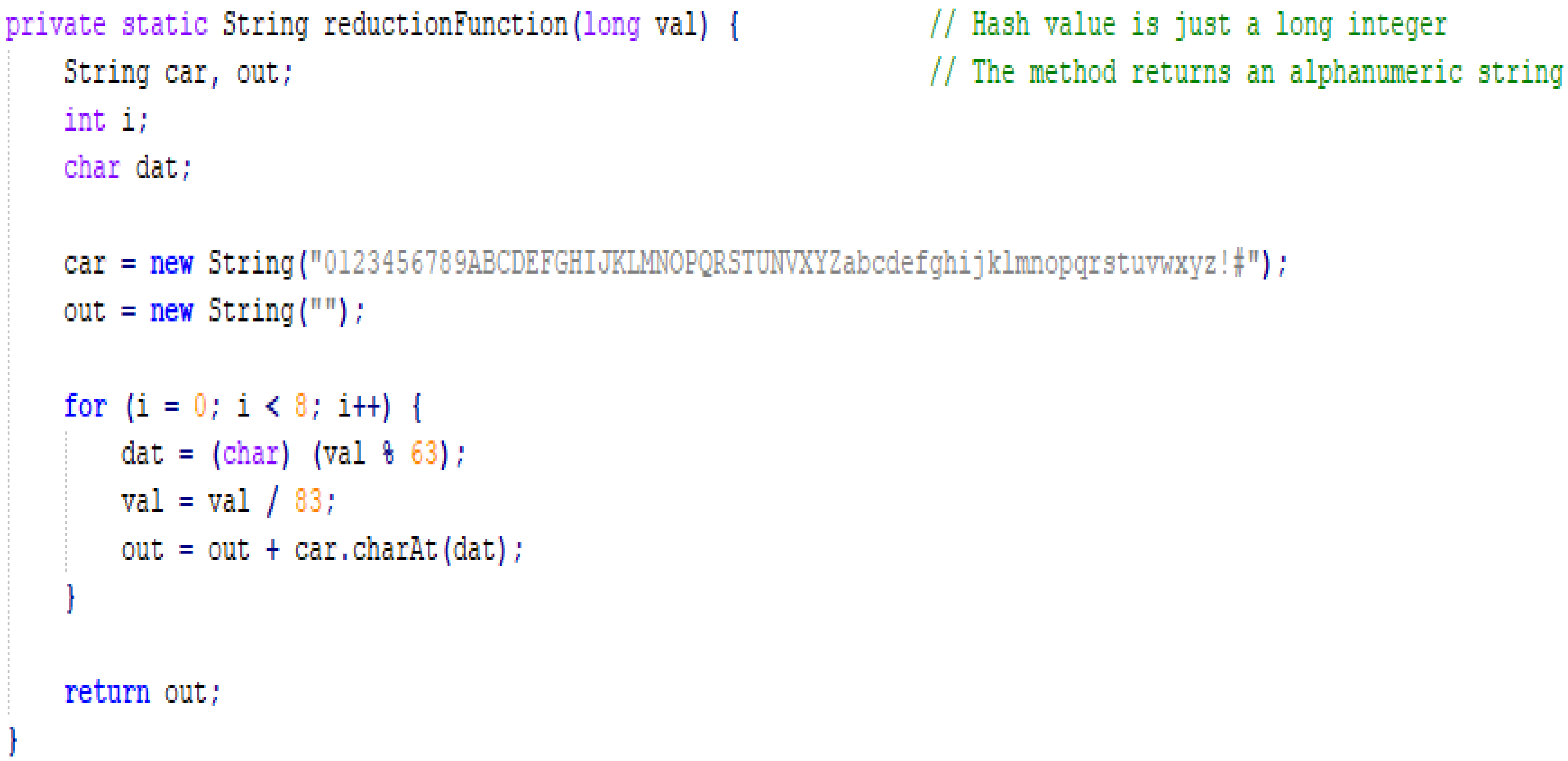
Rainbow Tables

- Look-up tables are huge and take up a lot of hard disk space
- Rainbow tables in contrast provide an efficient way to represent large numbers of hash values
- They require more processing time and less storage to find a match compared to a simple lookup table
- Rainbow tables are a practical example of a **space–time trade-off**
- They are based on pre-computed hash chains

Pre-Computed Hash Chains

- Such chains contain long sequences of password candidates (green strings below) and hash values (black strings below)
- The are based on using a hash function "→" and a reduction function "→", e.g.,
  aaaaaa→173bdfede2ee3ab3 → jdjkuo →9fdde3a0027fbb36 →… → k3rtol
  - In this example we only consider passwords (green) that are 6 characters long, which are converted into 64-bit hash values
  - Each chain starts with a different password
  - Each chain has a fixed length, e.g. 100,000 passwords and their hashes
  - Here "→" converts the 64-bit hash value into an arbitrary 6-byte long string again, i.e. it's not an inverted hash function!
- We only store the first and the last value (starting point and end point), i.e. "aaaaaa" and "k3rtol"

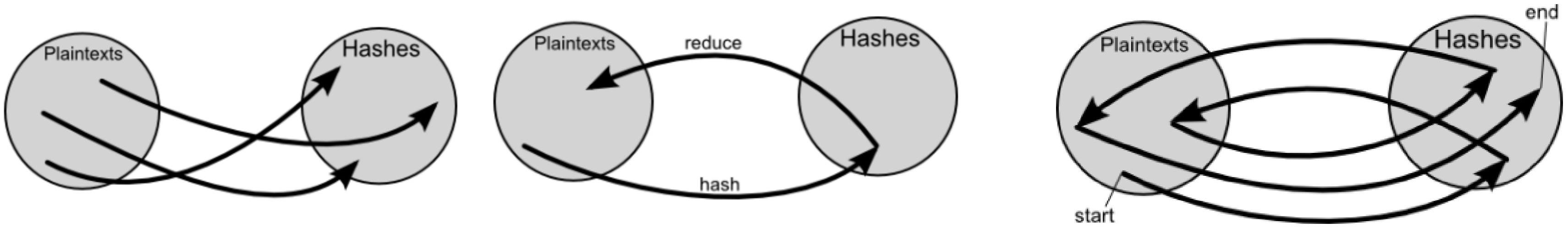
Example for a simple Reduction Function



```
private static String reductionFunction(long val) {          // Hash value is just a long integer
    String car, out;                                          // The method returns an alphanumeric string
    int i;
    char dat;

    car = new String("0123456789ABCDEFGHIJKLMNOPQRSTUNVXYZabcdefghijklmnopqrstuvwxyz!#");
    out = new String("");

    for (i = 0; i < 8; i++) {
        dat = (char) (val % 63);
        val = val / 83;
        out = out + car.charAt(dat);
    }

    return out;
}
```

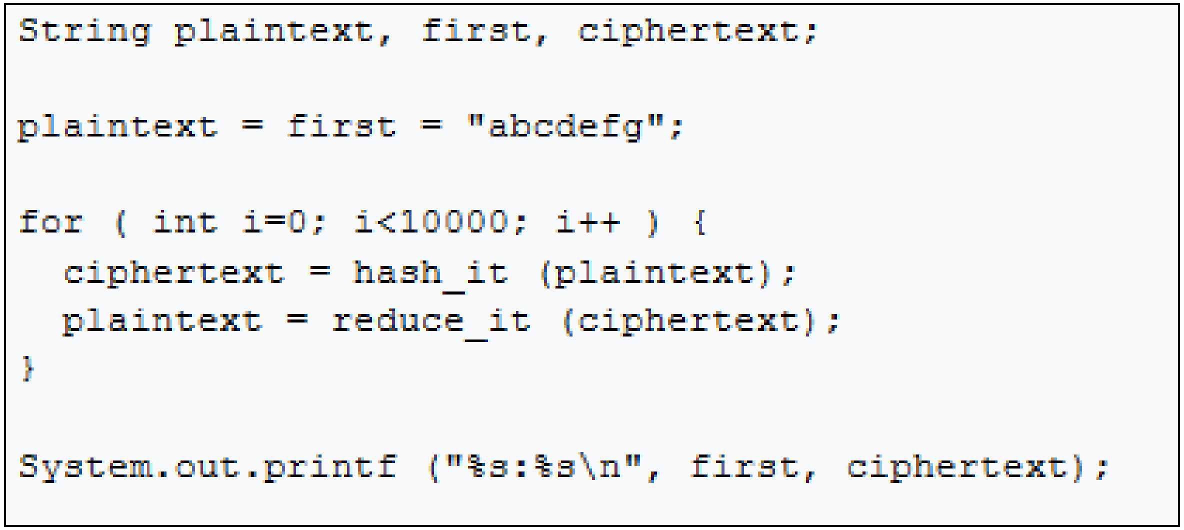
Coverage of Hash Chains

- The reduction function determines the range (i.e., length and composition) of plaintext (i.e., password) candidates that are covered
- Example:
  - Consider the password “Domino5”
  - In order to have this word stored in a chain, the reduction function must create outputs that are
    - At least 7 characters long
    - Contain small and capital letters, as well as numbers
  - Also, hash chains may not be able to cover all possible character combinations



Pseudo-Code to create a single Chain

- This example creates a chain with the start value “abcdefg” that covers 10,001 plaintext words
- Note that the last value of this chain is a hash value (i.e. ciphertext)
- We don’t know for certain what type of words the reduction function returns, possible only words of length 7 that consist of small letters only



```
String plaintext, first, ciphertext;

plaintext = first = "abcdefg";

for ( int i=0; i<10000; i++ ) {
  ciphertext = hash_it (plaintext);
  plaintext = reduce_it (ciphertext);
}

System.out.printf ("%s:%s\n", first, ciphertext);
```

Chain Lookup

Assume we have a table with just 2 chains (with start and end values), i.e.

aaaaaa→173bdfede2ee3ab3 →… → 8995tg →9fdde3a0027fbb36 →… → k3rtol
hfk39f→856385934954950 →… → delphi →759858fde66e8aa8 →… → prp56e

… and a hash value "759858fde66e8aa8" we'd like to crack

Starting with this hash value we apply consecutively "→" and "→", until we

- hit a known end value (e.g., k3rtol), or
- have repeated "→" and "→" x times (with x being the length of the chain)

If we hit a known end value, e.g. "prp56e", we repeat the transformation, beginning with the start value of the chain, i.e., "hfk39f", until we hit "759858fde66e8aa8" again

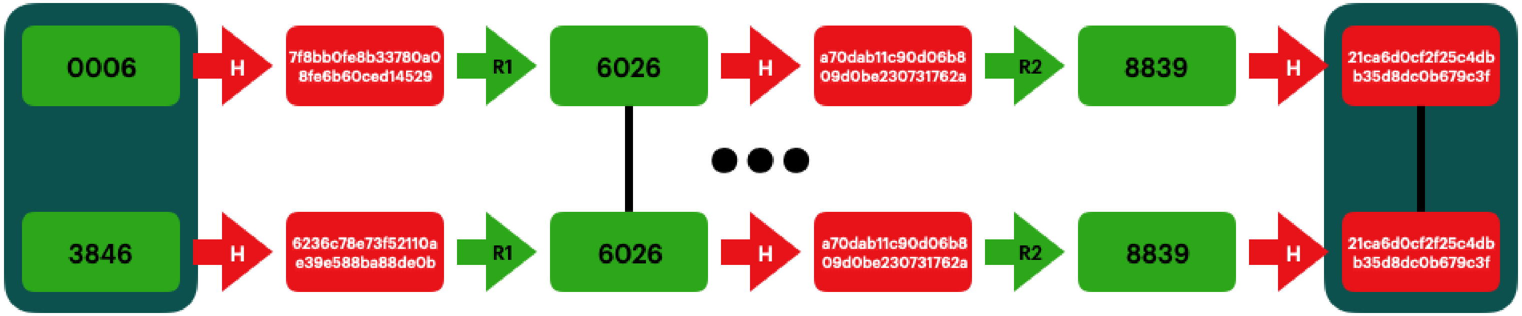
The input that led to the hash value (i.e., "delphi") is the solution

Chain Lookup Pseudocode

1. Input: Hash value H
2. Reduce H into another plaintext P
3. Look for the plaintext P in the list of final plaintexts (i.e. end values), if it is there, break out of the loop and goto step 6.
4. If it isn't there, calculate the hash H of the plaintext P
5. Goto step 2., unless you've done the maximum amount of iterations
6. If P matches one of the final plaintexts, you've got a matching chain; in this case walk through the chain in question again starting with the corresponding start value, until you find the text that translates into H

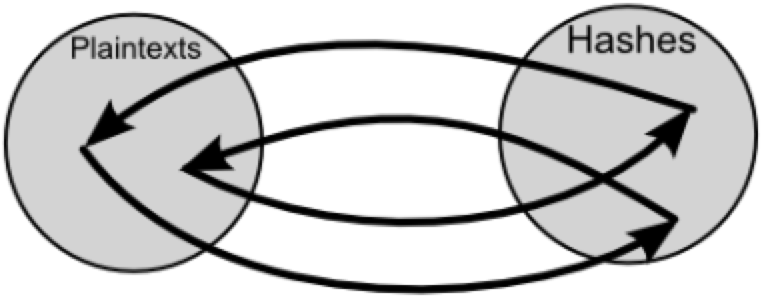
Chain Collisions

- Consider the following scenario:
  aaaaaa→ … → 173bdfede2ee3ab3 → delphi → 759858fde66e8aa8 →… → prp56e
  hfk39f→ … → 856385934954950 → delphi → 759858fde66e8aa8 →… → prp56e
- These 2 chains could merge, because
  - the reduction function translates two different hashes into the same password (as reduction functions are imperfect), or
  - the hash function translates two different passwords into the same hash (which should not happen → see hash function requirements)
- Because of these collisions or chain loops (next slide) hash chains will not cover as many passwords as theoretically possible despite having paid the same computational cost to generate
  - Previous chains are not stored in their entirety; therefore, it is impossible to detect this efficiently


0006
H
7f8bb0fe8b33780a0 8fe6b60ced14529
R1
6026
H
a70dab11c90d06b8 09d0be230731762a
R2
8839
H
21ca6d0cf2f25c4db b35d8dc0b679c3f
3846
H
6236c78e73f52110a e39e588ba88de0b
R1
6026
H
a70dab11c90d06b8 09d0be230731762a
R2
8839
H
21ca6d0cf2f25c4db b35d8dc0b679c3f

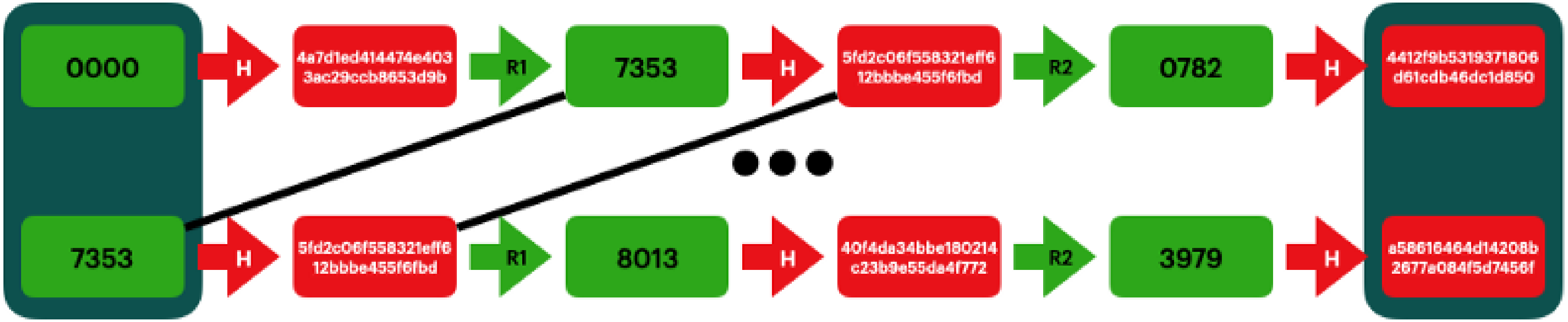
Chain Loops

- Here you find repetitions of hashes in a single chain
- The result of imperfect reduction functions that map two different hashes into the same plaintext

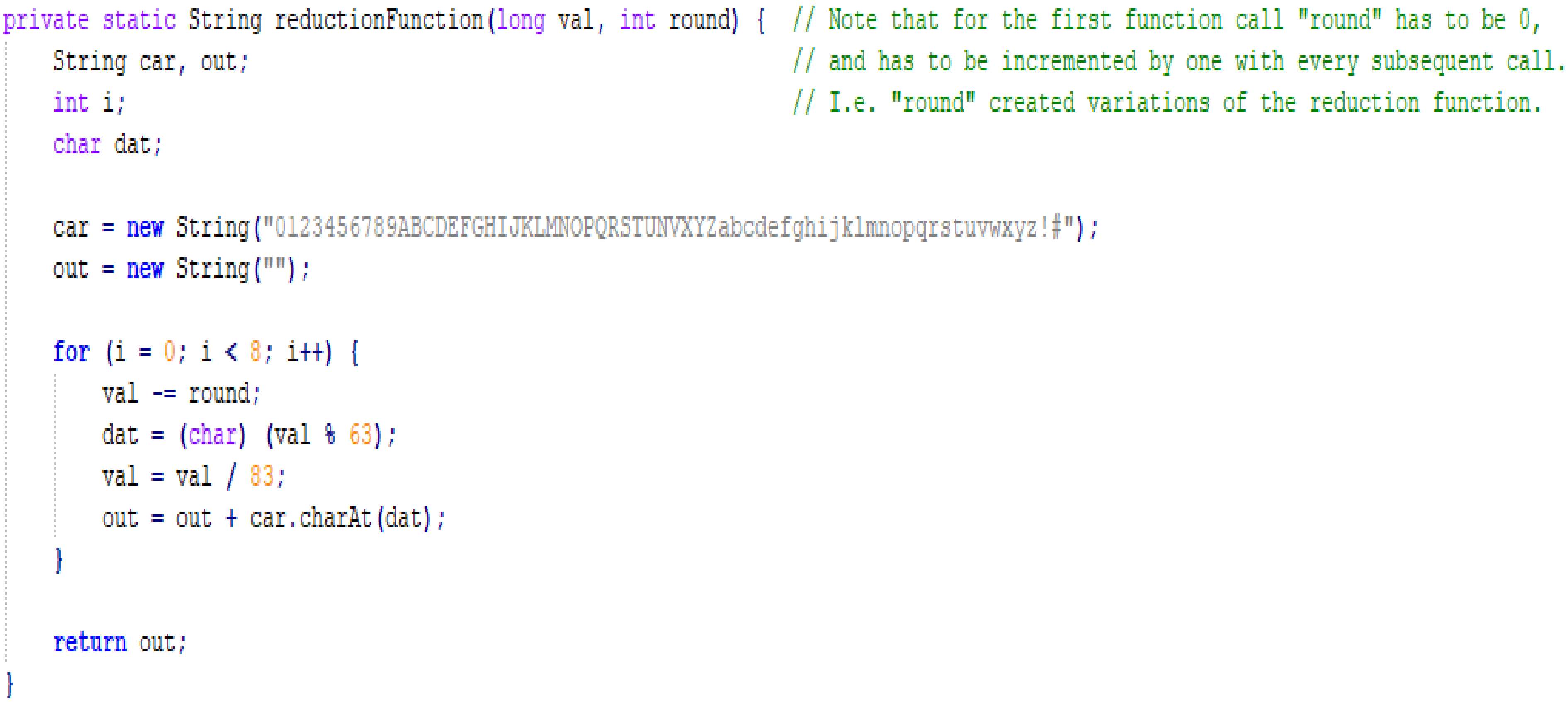


Rainbow Tables

- Rainbow tables effectively solve the problem of collisions with ordinary hash chains by replacing the single reduction function R with a sequence of related reduction functions R_1 through R_k (one reduction function per chain element)
- In this way, for two chains to collide and merge they must hit the same value on the same iteration, which is rather unlikely



Example for a Reduction Function for a Rainbow Table



```
private static String reductionFunction(long val, int round) {  // Note that for the first function call "round" has to be 0,
    String car, out;                                            // and has to be incremented by one with every subsequent call.
    int i;                                                      // I.e. "round" created variations of the reduction function.
    char dat;

    car = new String("0123456789ABCDEFGHIJKLMNOPQRSTUNVXYZabcdefghijklmnopqrstuvwxyz!#");
    out = new String("");

    for (i = 0; i < 8; i++) {
        val -= round;
        dat = (char) (val % 63);
        val = val / 83;
        out = out + car.charAt(dat);
    }

    return out;
}
```

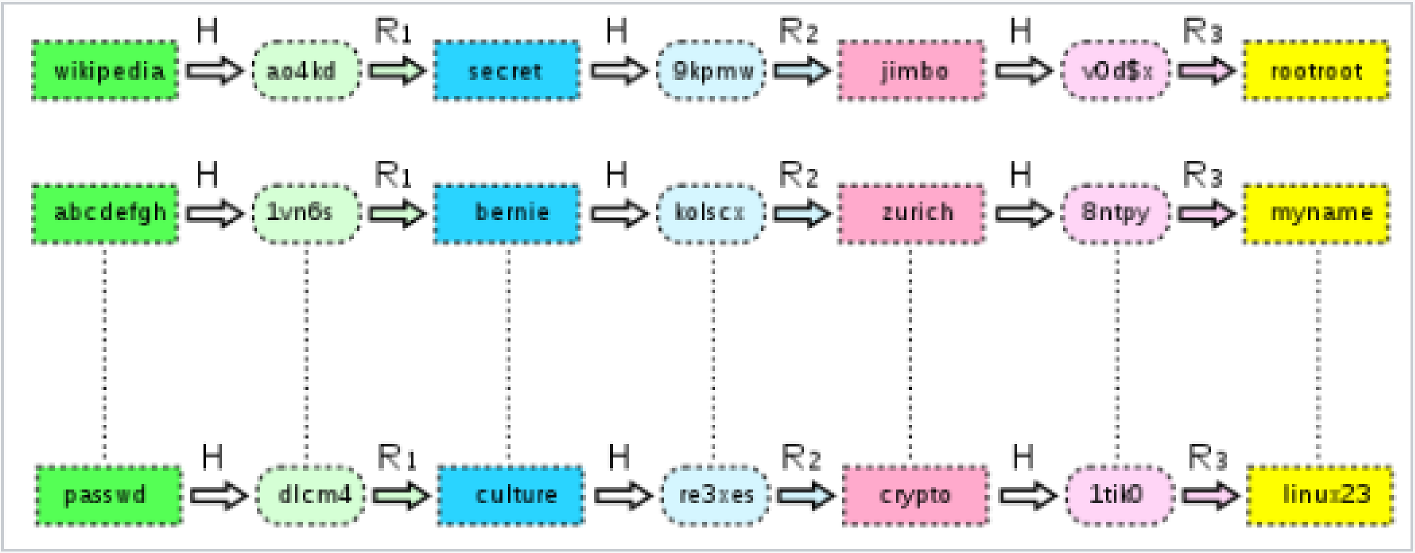
Coverage of Reduction Functions

- Rather than calculating a random string a reduction function may calculate an integer index value to identify an entry (word) in a large (password) dictionary
- Example:
  - H(lalo) = 368437FDA
  - R(368437FDA) = 6 ➔ dict[6] = robot123
  - H(robot123) = DDA0087e73
  - …
- This is similar to a lookup table, but requires far less space, as hashes are not stored
- However, it may be difficult to design a hash function that covers all dictionary indices

#	dict entry
0	Dog5
1	Simple
2	fEED2
3	lalo
4	mEn
5	hat
6	robot123
7	rose
…	

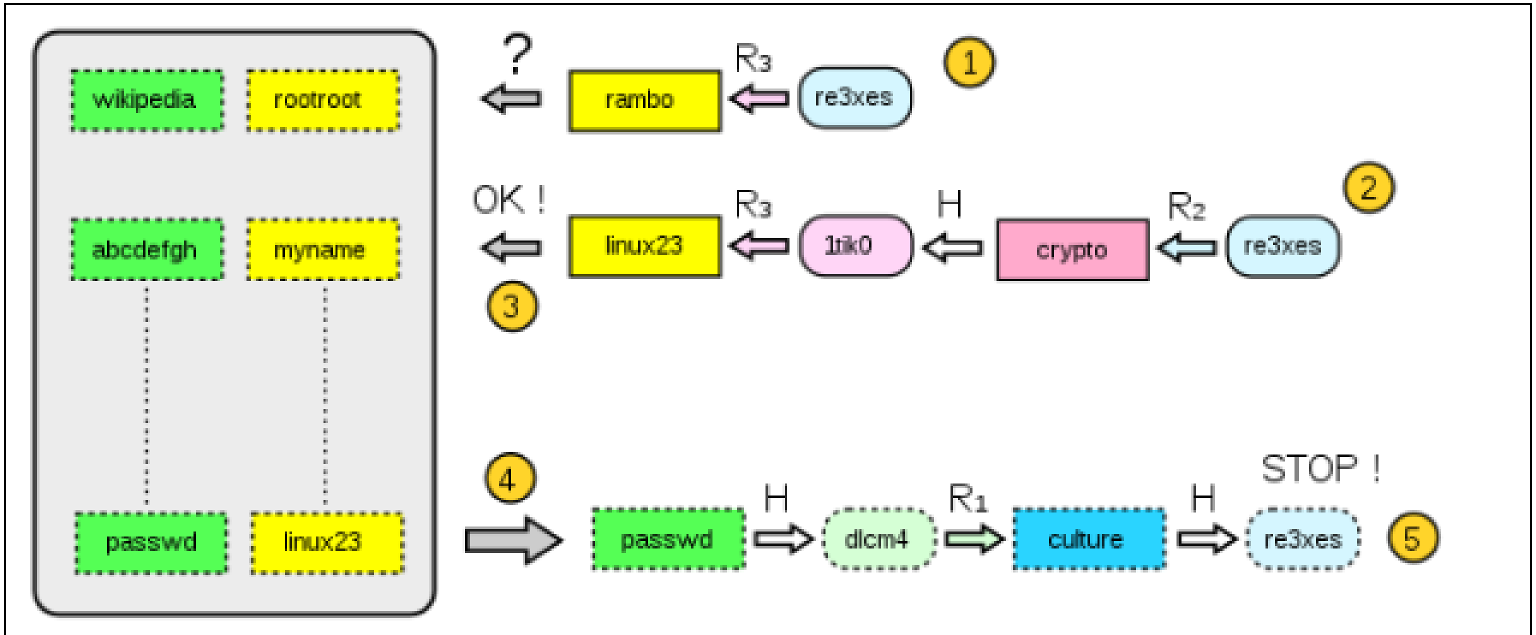
Searching a Rainbow Table (Wikipedia)

- □ Let's assume a Rainbow table of length 3 with 3 different reduction functions R_1, R_2 and R_3
- □ Again, we just store start (green) and end (yellow) value of each chain



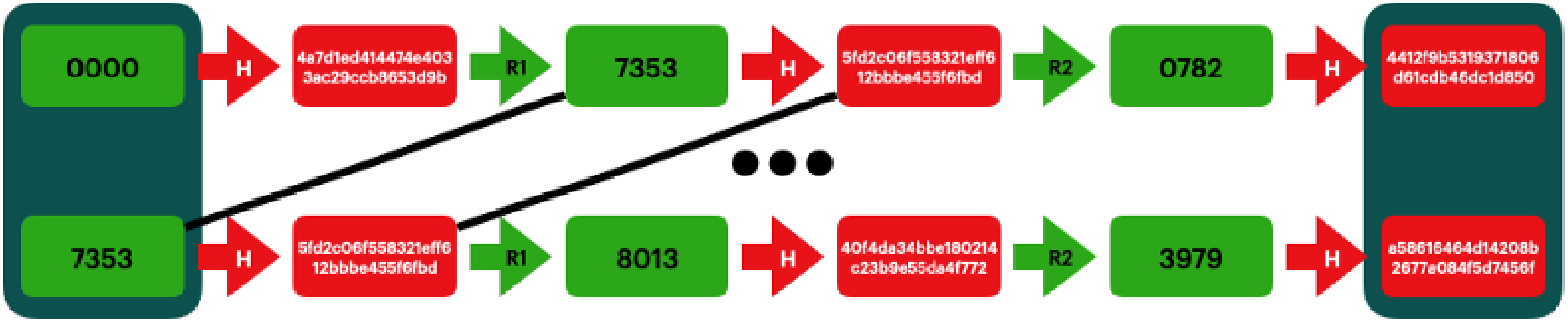
Searching a Rainbow Table (Wikipedia)

- Consider you have the rainbow table below and the password hash “re3xes”
  - Calculate R3(“re3xes”) and check if the result matches any of the chain ends (yellow boxes)
  - Calculate R2(H(R3(“re3xes”))) and check if the result matches any of the chain ends
  - ..
  - Repeat this process until the algorithm reaches R1, or a match is found
  - If a match is found, traverse through the chain in question as seen before, to find the solution



Perfect and non-perfect Rainbow Tables

- In a **perfect rainbow table** any word does not appear in more than one chain
- **Non-prefect rainbow tables (as shown below)** have redundant entries
  - They are easier to compute, but less memory-efficient because of these repetitions (which are not collisions!)



Defense against Rainbow Tables

- Idea:
  - Increase the (required minimum) length of a password
  - By doing so there are many more potential passwords to be considered by a rainbow table …
    - … up to a point where such tables are simply no more economical to generate
  - Increasing the password length can be either done by the
    - password owner (e.g., on the client side), or
    - algorithmically (e.g., on the client or server side)

Defence against Rainbow Tables

Client-side defence:

- A user requirement to choose long passwords that contain different types of characters,
  e.g. consider passwords that contain “A…Z”, “a…z”, “1-8”:
  - 6 characters long passwords result in 6^{60} = 46,656,000,000 combinations
  - 10 characters long passwords result in 10^{60} = 604,661,760,000,000,000 combinations

Server- (and potentially client-) side defence:

1. Password salting
   - A unique and random, but known string (“salt”) per user that is appended to each password before its hash is calculated
   - The salt is stored in the user database

User ID	Salt	Password Hash	Password (not part of table)
ms@gmail.com	12367	1d8922d005733…	12367KenSentme!
k51@outlook.com	56f87	628749afdb83…	56f87Fluffybear
abd@yahoo.com	465d0	980ade367fc93…	46d05Limerick

Defense against Rainbow Tables

2. Password peppering
   - Similar to Salting, but a unique ***secret*** string is concatenated to all passwords before they are hashed
4. Multiple iterations
   - A password is hashed multiple (e.g., 1000) times before stored in the database
5. Combination approach
   - Different techniques are combined to create a complex hash algorithm, e.g.,
   - NewHash(password) = hash(hash(password) || salt)

SQL Attacks

Some revision material covering

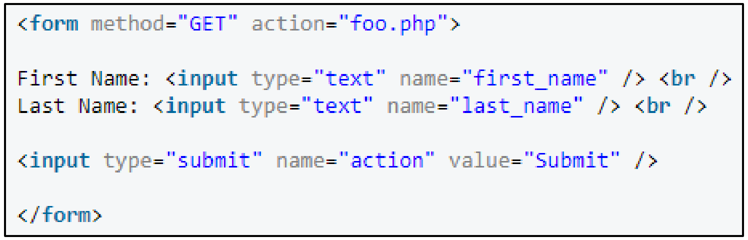
- SQL
- HTTP get / post Methods and PHP
- SQL injection attacks
- SQL injection attack mitigation strategies

What are SQL Injections?

- SQL injection is a code injection technique, used to attack data-driven applications, in which malicious SQL statements are inserted for execution
- A way of exploiting user input and SQL Statements to compromise the database and/or retrieve sensitive data
- Such attacks are closely linked to various web technologies, i.e. HTTP and PHP

HTTP get / post Methods and PHP

- PHP is a general-purpose server-side scripting language especially suited to web development
- PHP originally stood for Personal Home Page, but it now stands for the recursive initialism PHP: Hypertext Pre-processor
- The HTTP GET method sends the encoded user information appended to the page request
- The page and the encoded information are separated by the ? Character
- Example: http://www.test.com/index.htm?name1=value1&name2=value2
- PHP provides $_GET associative array to access all the sent information using GET method, e.g.



```
<form method="GET" action="foo.php">

First Name: <input type="text" name="first_name" /> <br />
Last Name: <input type="text" name="last_name" /> <br />

<input type="submit" name="action" value="Submit" />

</form>
```

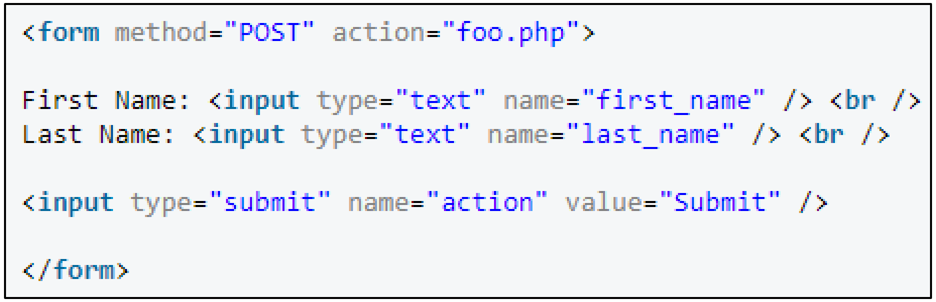
foo.php:

```
<?php
…
$var1 = $_GET['first_name'];
…
```

HTTP get / post Methods and PHP

- □ The POST method transfers information via HTTP headers
- □ The information is encoded as described in case of GET method and put into a header called QUERY_STRING
- □ The POST method does not have any restriction on data size and type to be sent
- □ The data sent by POST method goes through HTTP header (rather than the page request)
- □ PHP provides $_POST associative array to access all the sent information using POST method

```
foo.php:
<?php
…
 $var1 = $_POST[‘first_name’];
…
```



```
<form method="POST" action="foo.php">

First Name: <input type="text" name="first_name" /> <br />
Last Name: <input type="text" name="last_name" /> <br />

<input type="submit" name="action" value="Submit" />

</form>
```

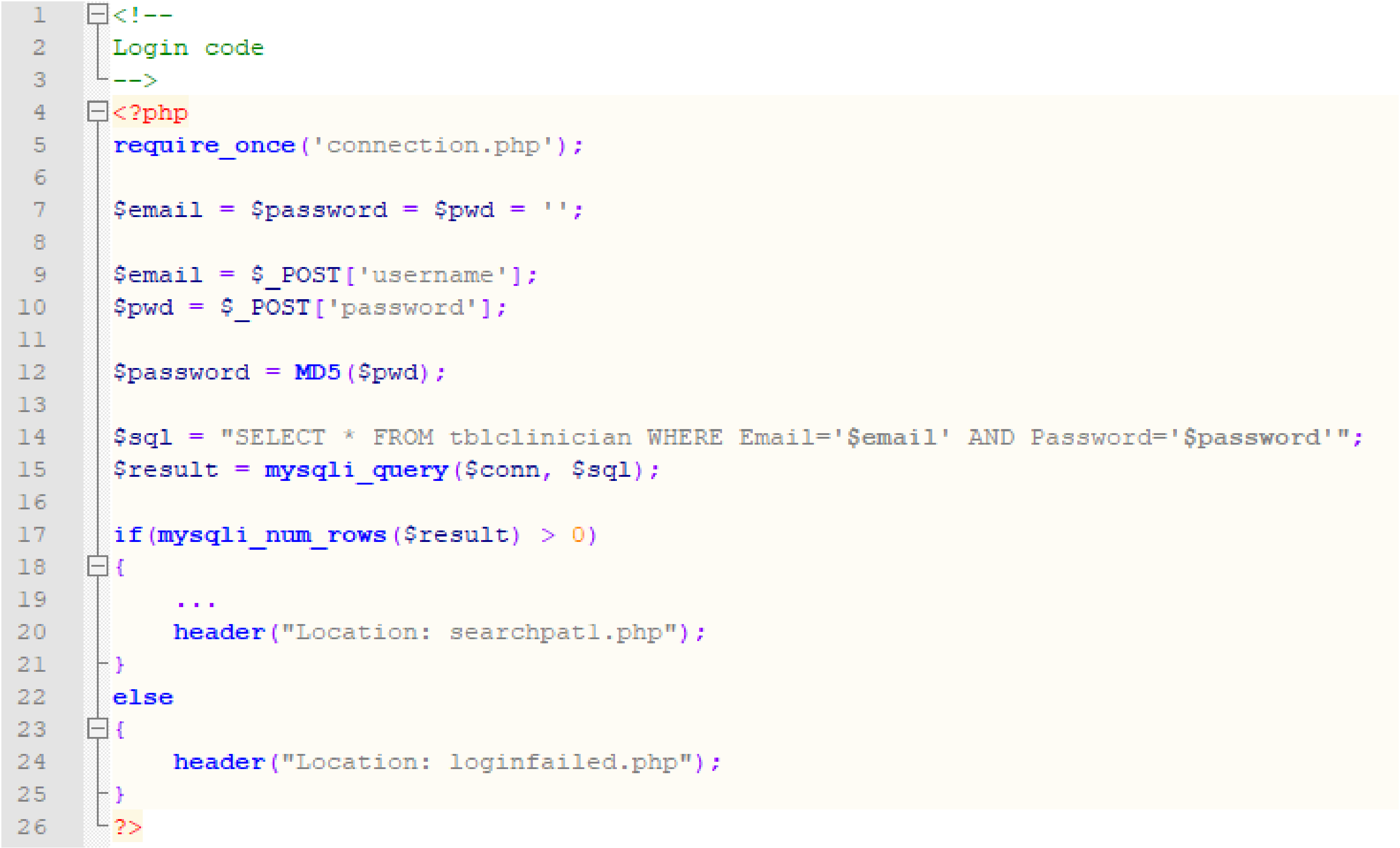
SQL Syntax Review

- Basic select query:
  SELECT <columns> FROM <table> WHERE <condition>
- Example:
  SELECT * FROM user WHERE id = 1 AND pass = 'bla'
- Note:
  - Literal strings are delimited with single quotes
  - Numeric literals aren't delimited

SQL Syntax Review

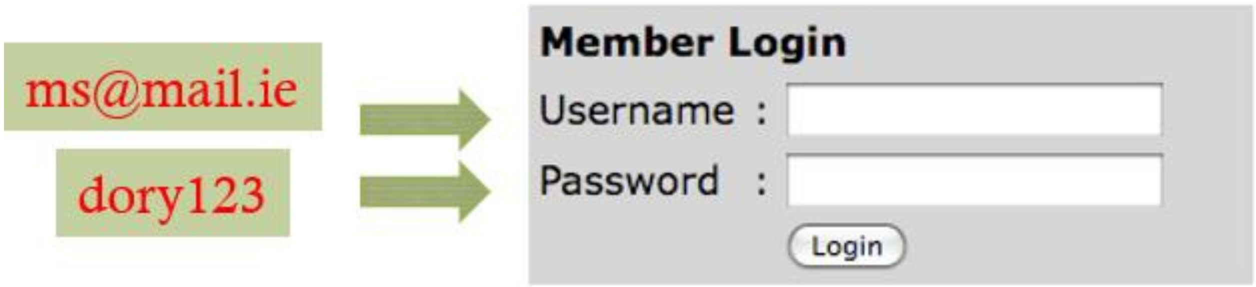
- Some databases allow semicolons to separate multiple statements:
  DELETE FROM user WHERE id = 1; INSERT INTO user (id, pass) VALUES (1, 'secure');
- For most SQL variants, the sequence -- means the rest of the line should be treated as a comment

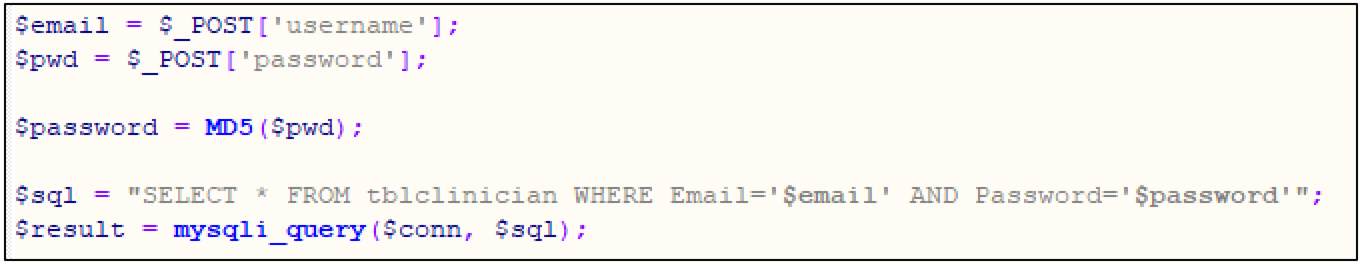
SQL Code Injection Example



```
<!--
Login code
-->
<?php
require_once('connection.php');

$email = $password = $pwd = '';

$email = $_POST['username'];
$pwd = $_POST['password'];

$password = MD5($pwd);

$sql = "SELECT * FROM tblclinician WHERE Email='$email' AND Password='$password'";
$result = mysqli_query($conn, $sql);

if(mysqli_num_rows($result) > 0)
{
    ...
    header("Location: searchpatl.php");
}
else
{
    header("Location: loginfailed.php");
}
?>
```

SQL Code Injection Example





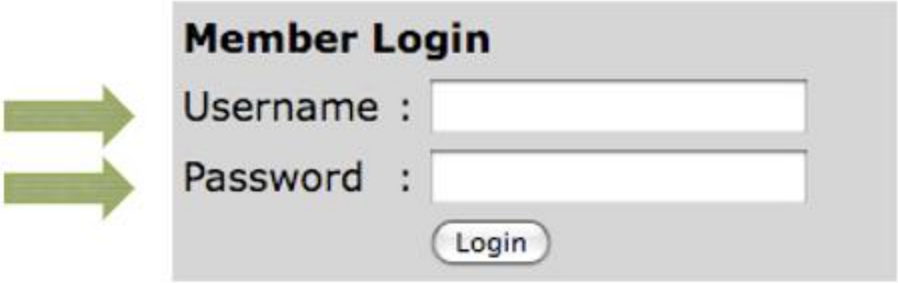
```
$email = $_POST['username'];
$pwd = $_POST['password'];

$password = MD5($pwd);

$sql = "SELECT * FROM tblclinician WHERE Email='$email' AND Password='$password'";
$result = mysqli_query($conn, $sql);
```

Table tblclinician:

Email	Hashed Password
ms@mail.ie	af47f8d1ac4
…	…

SQL Code Injection Example

‘; DROP TABLE tblclinician; --



Member Login

Username :

Password :

Login

$sql = "SELECT * FROM tblclinician WHERE Email=“; DROP TABLE tblclinician; --’ AND Password=“

- □ Note: The SQL DROP TABLE statement deletes an existing table in a database
- □ While an attacker does not know the tables’ names, the attacker can do a **blind attack**
- □ More generally, If DB details are not known to the attacker, **blind SQL injections** are used

Other Code Injections if DB structure is known

- □ SELECT * FROM tblclinician WHERE Email =“; INSERT INTO tblclinician (Email,Password) VALUES (‘hacker’,123);--’ AND `Password`=“

- □ SELECT * FROM `login` WHERE Email =“; UPDATE tblclinician SET Password = 1284ffa WHERE Email = ms@mail.ie ;--’ AND `Password`=“

- □ The first injection creates a new user (hacker) including password hash
- □ The second injection replaces a user’s password hash

Types of SQL Injection Attacks

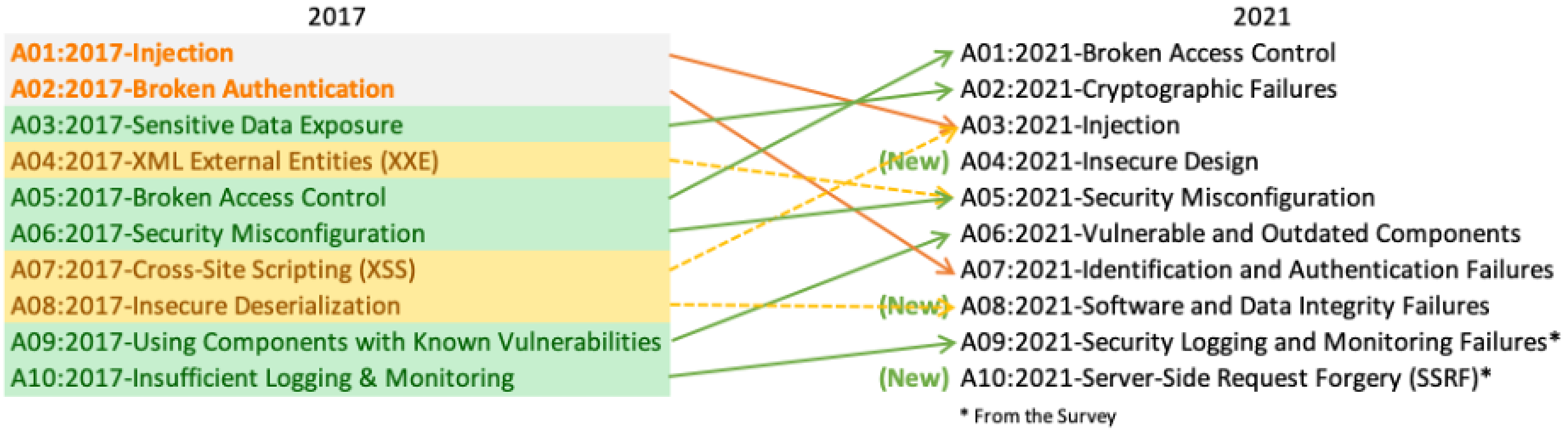
- **Blind SQL Injection**
  - Enter an attack on one vulnerable page but it may not display results
  - A second page would then be used to view the attack results
- **Conditional Response**
  - Test input conditions to see if an error is returned or not
  - Depending on the response, the attacker can determine yes or no information
- **First Order Attack**
  - Runs right away
- Second Order Attack
  - Injects data which is then later executed by another activity (job, etc.)
- Lateral Injection
  - Attacker can manipulate values using implicit functions

What is at Risk?

- Any web application that accepts user input
  - Both public and internal facing sites
  - Public facing sites will likely receive more attacks than internal facing sites
- For the last couple of years (i.e. since 2013), (SQL) Injection is one of the frontrunners on the OWASP top ten list
  - A well understood attack, but still not fully grasped by the developer community

OWASP Top 10

- The Open Web Application Security Project (OWASP) is a non-profit foundation dedicated to improving the security of software



2017

A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring

2021

A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*

* From the Survey

Some historical Notes

- Guess Inc. is an American clothing brand and retailer
- Guess.com was open to a SQL injection attack
- In 2002 Jeremiah Jacks discovered the hole and was able to pull down 200,000 names, credit card numbers and expiration dates in the site's customer database
- The episode prompted a year-long investigation by the US Federal Trade Commission

Some historical Notes

- In 2003 JJ used an SQL injection to retrieve 500,000 credit card numbers from PetCo
- In 2014 Russian hackers used a Botnet to recover a vast collection of stolen data, including 1.2 billion unique username/password pairs, by compromising over 420,000 websites using SQL injection techniques

What can SQL Injections do?

- Retrieve sensitive information, including
  - Usernames/ **Passwords**
  - Credit Card information
  - Social Security / PPS numbers
- Manipulate data, e.g.
  - Delete records
  - Truncate tables
  - Insert records
- Manipulate database objects, e.g.
  - Drop tables
  - Drop databases

What can SQL Injections do?

- Retrieve System Information
  - Identify software and version information
  - Determine server hardware
  - Get a list of databases
  - Get a list of tables
  - Get a list of column names within tables
- Manipulate User Accounts
  - Create new sysadmin accounts
  - Insert admin level accounts into the web-app
  - Delete existing accounts

CT437
COMPUTER SECURITY AND FORENSIC COMPUTING

PUBLIC KEY CRYPTOGRAPHY

Dr. Michael Schukat

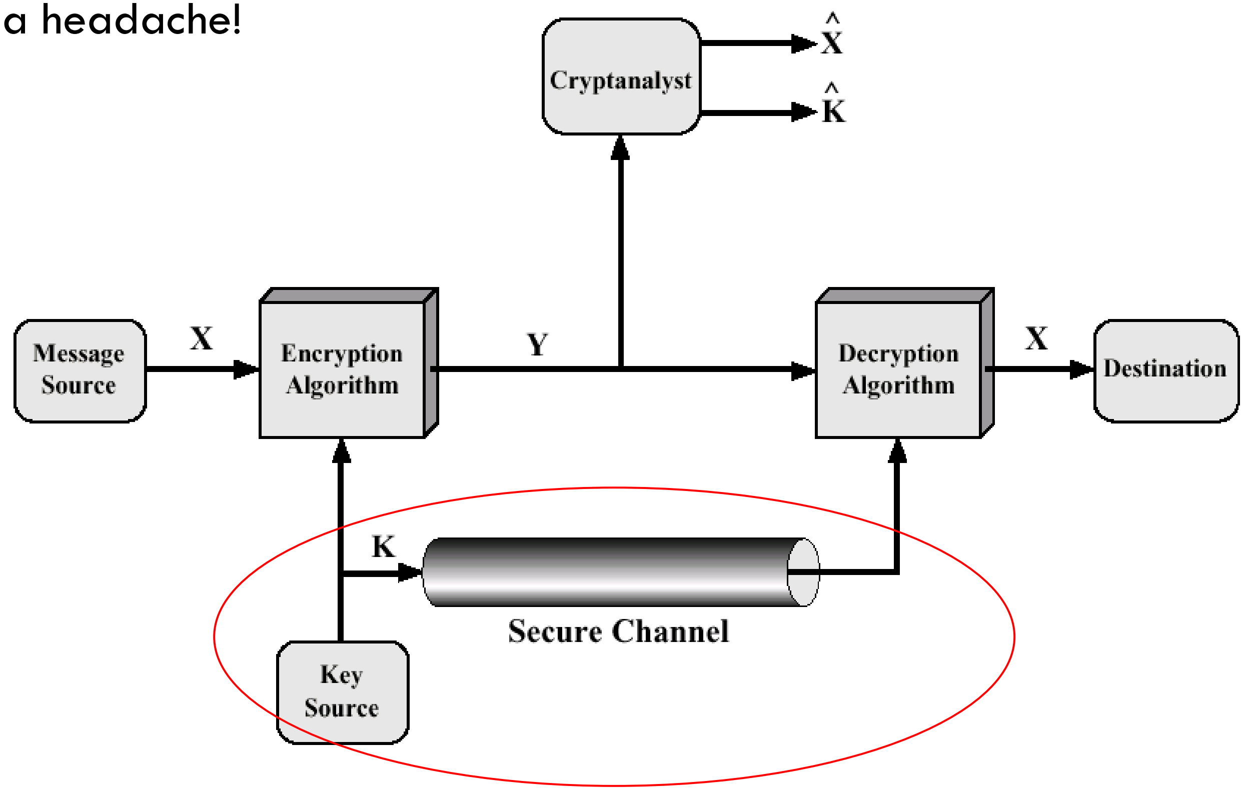
Ollscoil na Gaillimhe
University of Galway

Lecture Content

- Public key cryptography versus private key cryptography
- Public key cryptography applications
- Diffie-Hellman Key exchange
  - Man-in-the-Middle (MitM) attacks
- RSA encryption
- Optimisation techniques for public key encryption
- ECC encryption
- The Double-Ratchet algorithm

Model of Conventional Cryptosystem

Symmetric block ciphers are cryptographically strong, but key distribution can be a headache!



$$Y = E_K(X),\ X = E_K^{-1}(Y)$$

Features and Limitations of Private-Key Cryptography

- Traditional symmetric/single key cryptography uses one key, shared by both sender and receiver
  - If this key is disclosed, communications are compromised

- The key is also symmetric, both parties are equal
  - This is problematic too, as it does not protect the sender from a situation, where:

    - the receiver forges a message using that key

    - and claims that it was sent be the sender

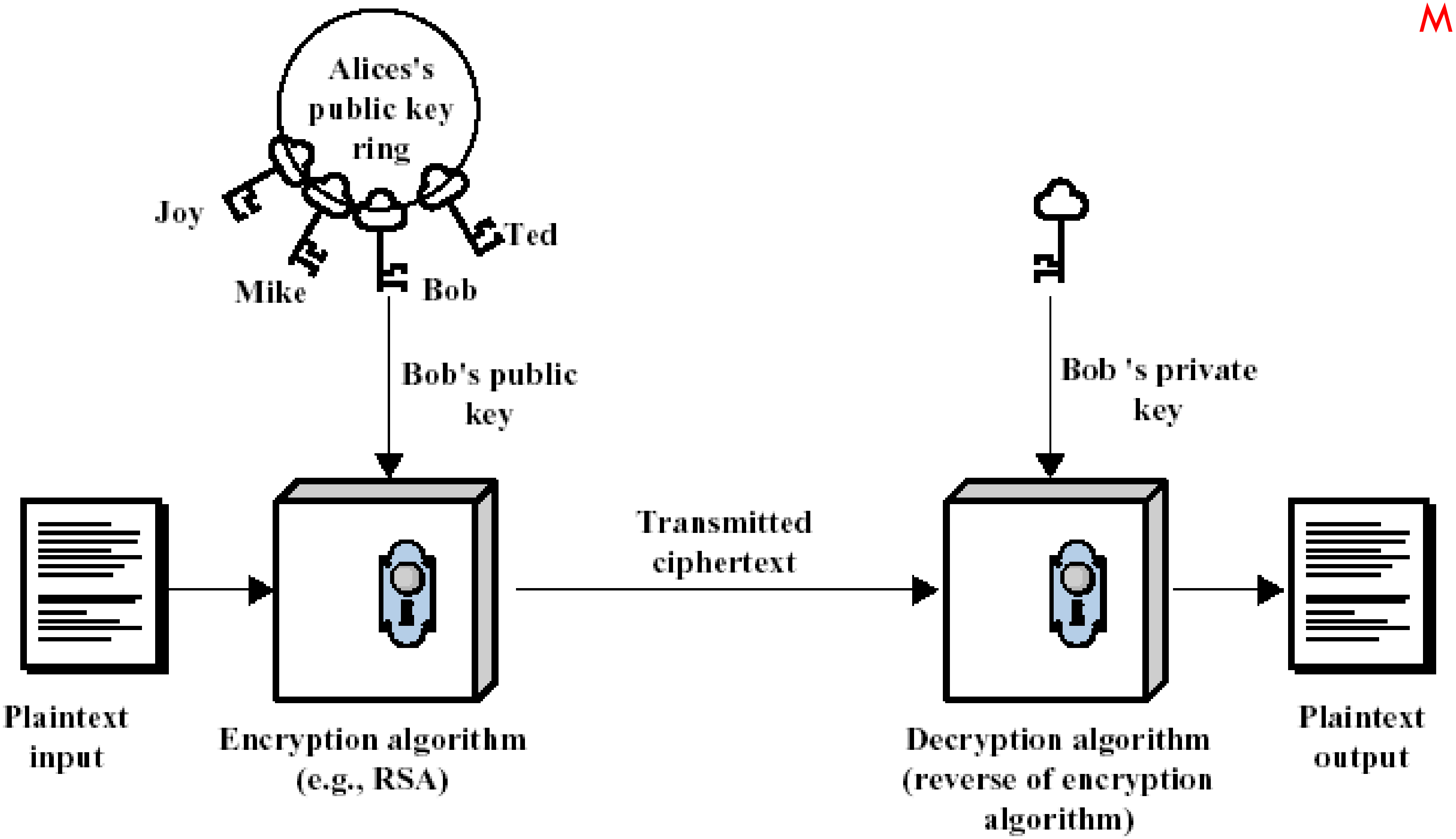
    - Think about an electronic contract that is exchanged between two business partners that use a shared key
    - One party can forge a contract and claim it was sent by the other side
    - Message authentication (HMAC or CMAC) doesn't solve the problem!

Features of Public-Key Cryptography

- **Public-key/two-key/asymmetric cryptography** involves the use of two keys:
  - a **public-key**, which the owner shares with any peer; it is used to:
    - Encrypt messages send from the peer to the owner
    - Verify the integrity and origin of messages send from the owner to a peer (signature validation)
  - a **private-key**, known only to the recipient/owner, used to:
    - Decrypt messages that were encoded using their public key
    - Digitally sign data send to a peer (signature creation)
- The keys are **asymmetric**, because they are not equal
- Those who encrypt a message or verify a signature (using the receiver's public key) cannot decrypt the message or forge a signature
- It is computationally very hard (and infeasible) for an attacker to rebuild an owner's private key by analysing their public key
- This is achieved through the application of number- theoretic concepts

Public-Key Encryption

M

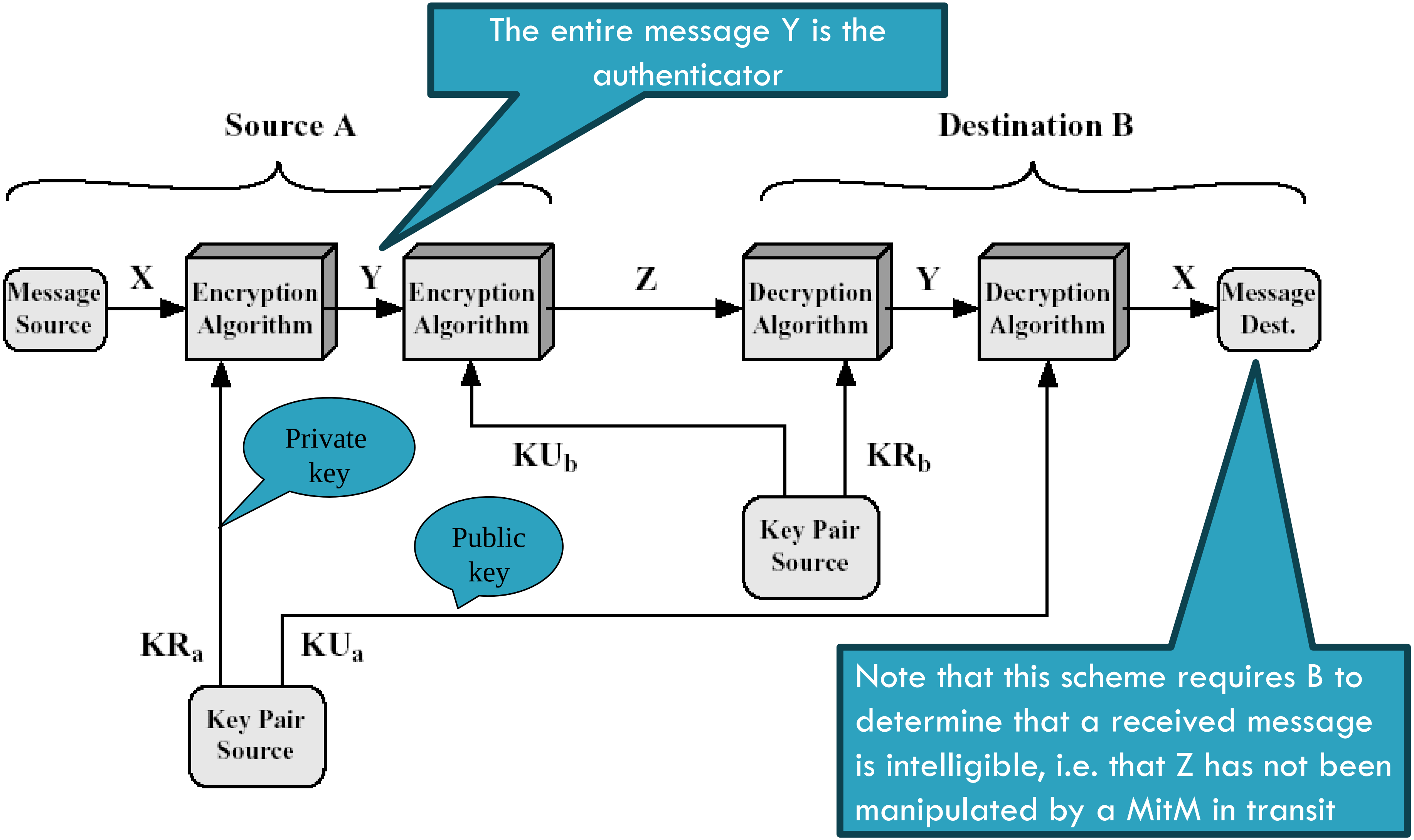


Applications of Public-Key Cryptosystems

- **Data encryption/decryption:**
  The sender encrypts the message with the recipient's public key and the receiver decodes the message using their private key
  - Recall symmetric encryption where only **one** key is used
- **Digital signature/authentication:**
  The sender "signs" a message with their private key. Signing is achieved by encrypting the message or its MAC using their private key (next slide)
  - Recall private key encryption where sender and receiver just share one key
- **Key exchange:**
  Two sides negotiate a **symmetric** session key
  - Private key encryption is much faster than public key encryption
  - This key may also be used for conventional message authentication

- Note that in order to avoid confusion we use from now on the terms:
  - Symmetric key for private key encryption (block ciphers and stream ciphers)
  - Public and private keys for public key encryption

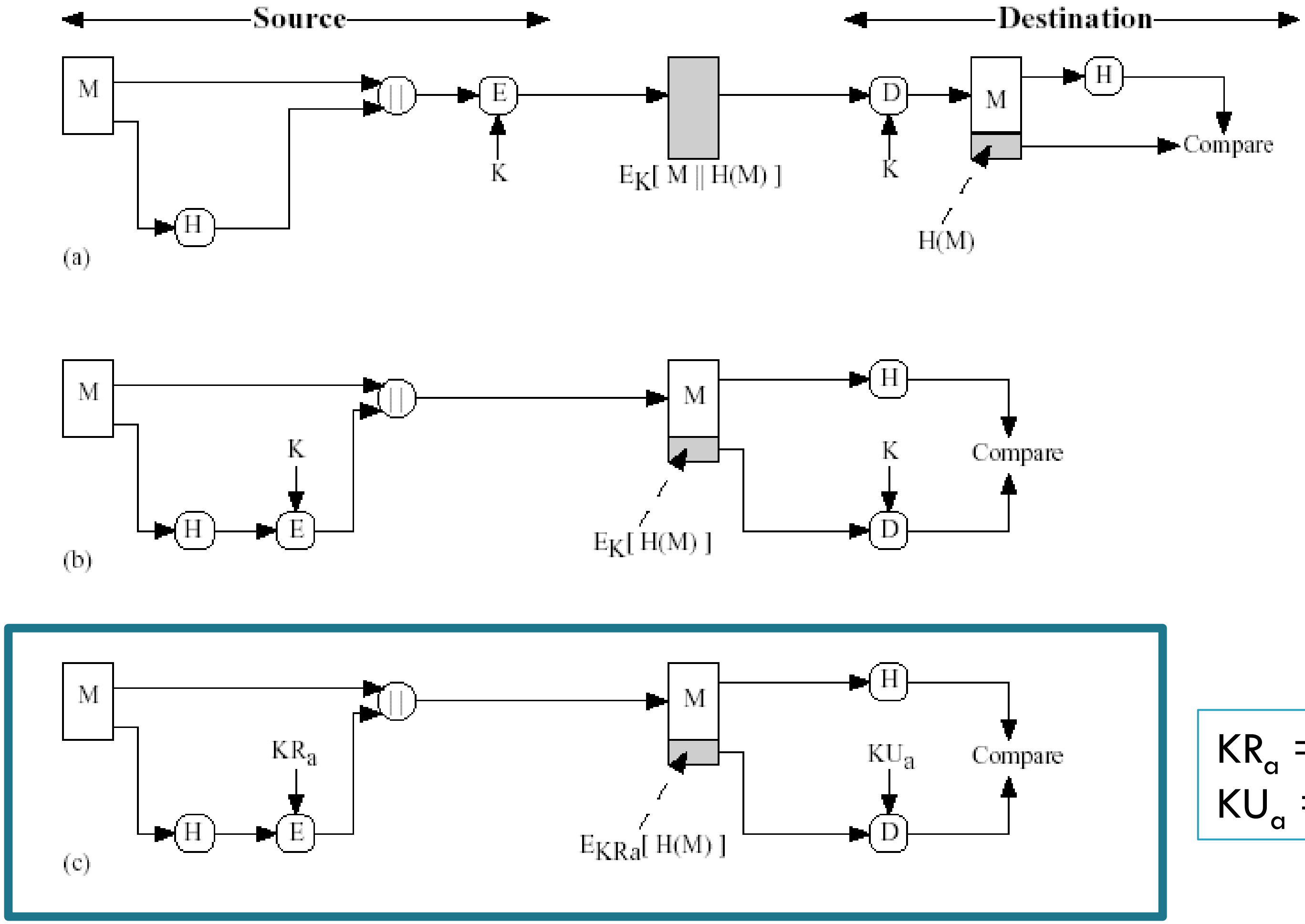
Public-Key Cryptosystems: Secrecy and Authentication



The entire message Y is the authenticator

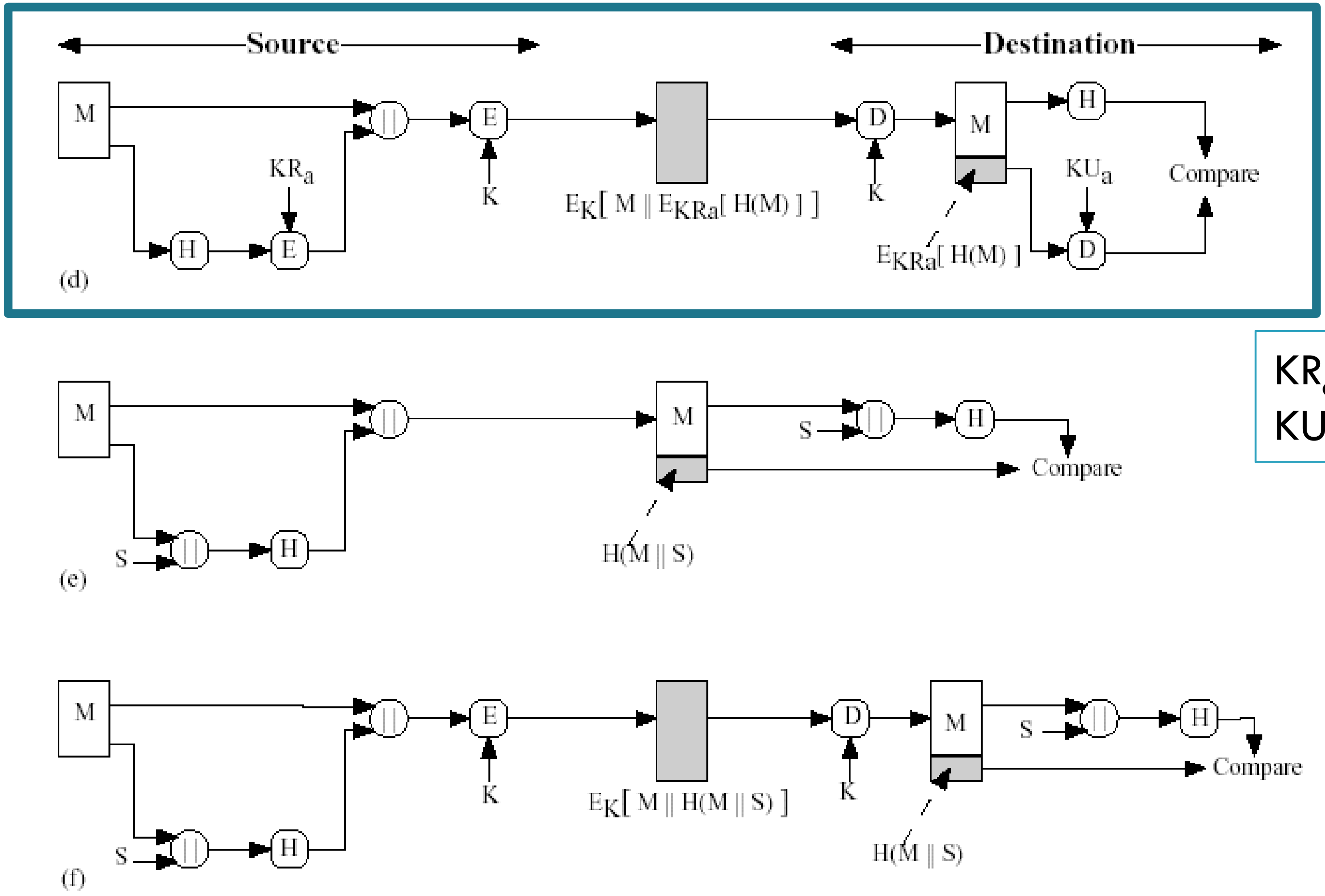
Note that this scheme requires B to determine that a received message is intelligible, i.e. that Z has not been manipulated by a MitM in transit

Recap: Basic Uses of Hash Functions (H) in Combination with asymmetric Encryption (c)



KR_a = Sender’s private key
KU_a = Sender’s public key

Recap: Basic Uses of Hash Functions (H) in Combination with asymmetric Encryption (d)

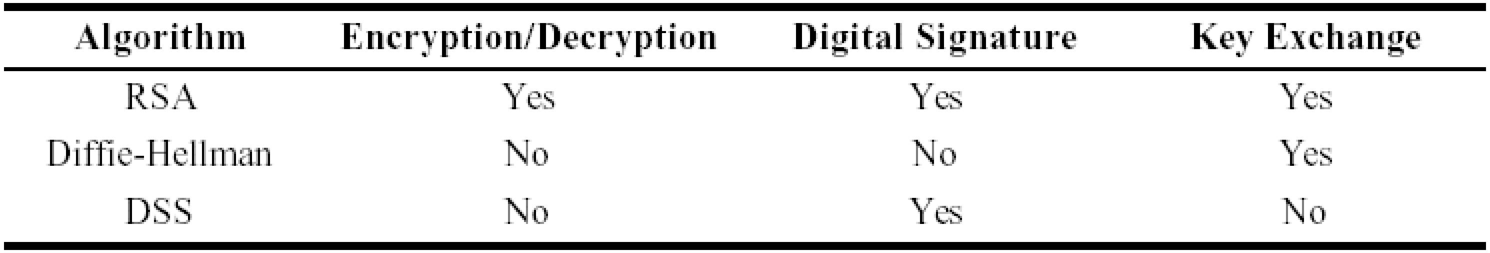


KR_a = Sender’s private key

KU_a = Sender’s public key

Public-Key Cryptosystems

- There are different cryptosystems, including (from simplest to most complex):
  - **Diffie Hellman key exchange**
  - **RSA**
  - DSS
  - **Elliptic Curve Cryptography**



Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

Modular Arithmetic

- Modular arithmetic is a system of arithmetic for integers, where numbers wrap around when reaching a certain value n, called the modulus
  - Recall modulus operator “%” in C and other languages, i.e. “division with rest” with rest being the modulus
  - Example: 75 / 6 = 12 remainder 3 → 75 % 6 = 3
- Numbers {0, 1, …, n - 1} are called “multiplicative group of integers modulo n”, or simply Z_n, for some n > 0
- Within Z_n, addition and multiplication is well defined!

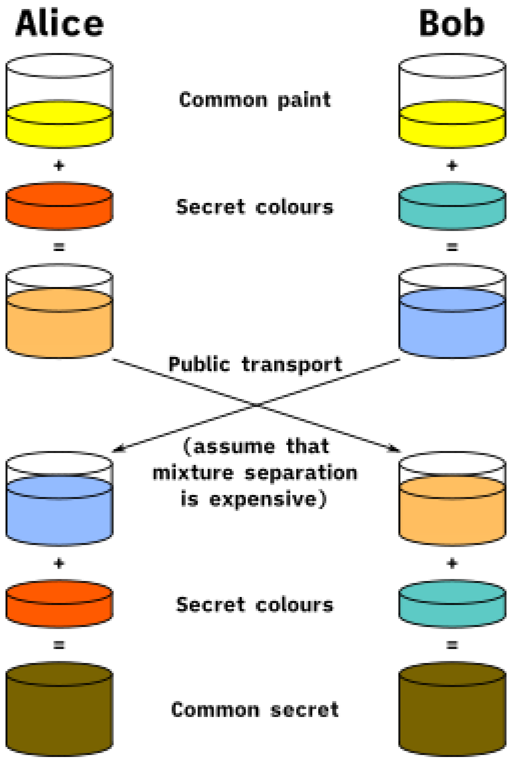
Example: Multiplication in Z_9

Mx3

*	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8
2	0	2	4	6	8	1	3	5	7
3	0	3	6	0	3	6	0	3	6
4	0	4	8	3	7	2	6	1	5
5	0	5	1	6	2	7	3	8	4
6	0	6	3	0	6	3	0	6	3
7	0	7	5	3	1	8	6	4	2
8	0	8	7	6	5	4	3	2	1

Illustration of Concept behind Diffie-Hellman Key Exchange (Wikipedia)

- Alice and Bob want to share a secret colour using public transport
  - i.e. an adversary (i.e. Mallory, not shown) can get samples of any colour that is exchanged between both
- Alice and Bob agree on a common "public" paint color (yellow in the example)
- Each of them add a secret colour and send their mix to the other party
  - Mallory can intercept both, but cannot separate the mixtures
- Alice and Bob receive the other's mixture and add their secret colour
- Both colours are identical
- → This color is their common secret



Diffie-Hellman Key Exchange

- Diffie-Hellman provides a mechanism for a secure key exchange between two endpoints
  - The negotiated key is subsequently used as a symmetric key (or as a seed for a key) for data encryption and message authentication (as seen before)
- The algorithm uses the multiplicative group of integers modulo q
  - q has typically a length of 1024 or 2048 bits
- It is based on the difficulty of computing discrete logarithms over such groups, e.g.

$$6^{3} \bmod 17 = 216 \bmod 17 = 12 \qquad \text{(easy)}$$

$$12 = 6^{y} \bmod 17? \qquad \text{(difficult)}$$

  - Recall $6^3 = 6 \times 6 \times 6$, so we need just the multiplication

- The core equation for the key exchange is

$$K = (A)^{B} \bmod q$$

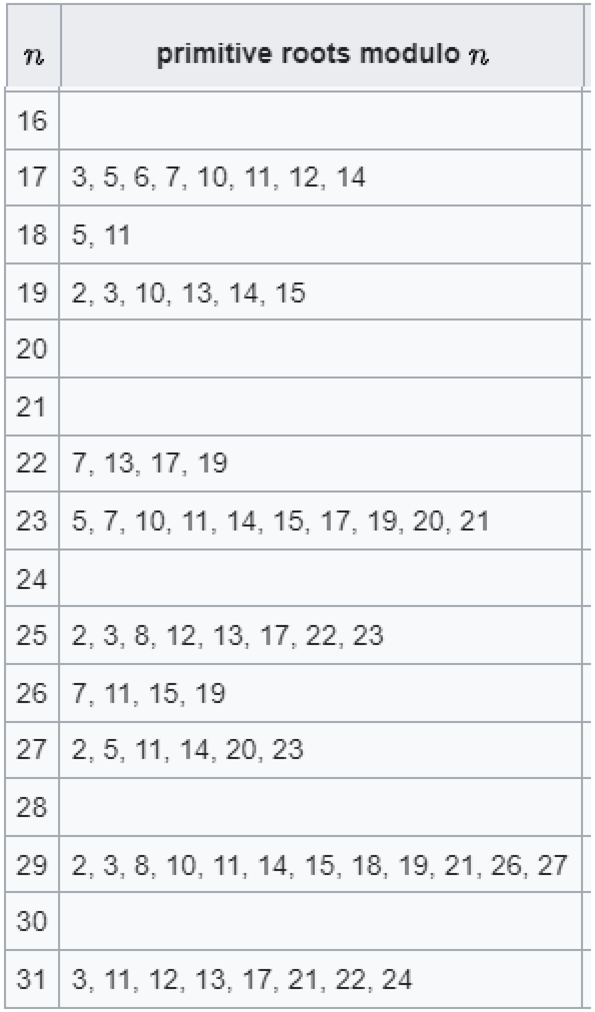
Diffie-Hellman: Global Public Elements

- Alice and Bob select:
  - A prime number q which determines Z_q
  - A positive integer a, with $1 < a < q$ and a is a **primitive root** of q
    - Note that a is also called the generator
- Definition: a is a primitive root of q, if numbers
  $a \bmod q,\ a^2 \bmod q,\ \cdots\ a^{(q-1)} \bmod q$
  are distinct integer values between 1 and $(q - 1)$ (i.e. in Z_q) in some permutation

- Example: $a = 3$ is a primitive root of Z_5 (i.e. $q = 5$), $a = 4$ is not:

$3^1 = 3 = 0 * 5 + \mathbf{3}$	$4^1 = 4 = 0 * 5 + \mathbf{4}$
$3^2 = 9 = 1 * 5 + \mathbf{4}$	$4^2 = 16 = 3 * 5 + \mathbf{1}$
$3^3 = 27 = 5 * 5 + \mathbf{2}$	$4^3 = 64 = 12 * 5 + \mathbf{4}$
$3^4 = 81 = 16 * 5 + \mathbf{1}$	$4^4 = 256 = 51 * 5 + \mathbf{1}$

Primitive Roots of Z_n with 15 < n < 32



n	primitive roots modulo n
16	
17	3, 5, 6, 7, 10, 11, 12, 14
18	5, 11
19	2, 3, 10, 13, 14, 15
20	
21	
22	7, 13, 17, 19
23	5, 7, 10, 11, 14, 15, 17, 19, 20, 21
24	
25	2, 3, 8, 12, 13, 17, 22, 23
26	7, 11, 15, 19
27	2, 5, 11, 14, 20, 23
28	
29	2, 3, 8, 10, 11, 14, 15, 18, 19, 21, 26, 27
30	
31	3, 11, 12, 13, 17, 21, 22, 24

Generation of Secret-Key: Part 1

- Alice and Bob share publicly a prime number q and a primitive root a
- Alice (User A):
  - Select secret number XA with 0 < XA < q
  - Calculate public value $YA = a^{XA} \bmod q$ (← difficult to reverse)
  - YA is sent to Bob (user B)
- Bob (User B):
  - Select secret number XB with 0 < XB < q
  - Calculate public value $YB = a^{XB} \bmod q$ (← difficult to reverse)
  - YB is send to Alice

Generation of Secret-Key: Part 2

- Alice:
  - Alice owns XA and receives YB
  - She generates the secret key: $K = (YB)^{XA} \bmod q$
- Bob:
  - Bob owns XB and receives YA
  - Bob generates the secret key: $K = (YA)^{XB} \bmod q$
- **Both keys are identical!**

Generation of Secret-Key: Part 2

$$K = (YB)^{XA} \bmod q$$

$$= (a^{XB} \bmod q)^{XA} \bmod q$$

$$= (a^{XB})^{XA} \bmod q$$

$$= a^{XB\ XA} \bmod q$$

$$= a^{XA\ XB} \bmod q$$

$$= (a^{XA})^{XB} \bmod q$$

$$= (a^{XA} \bmod q)^{XB} \bmod q$$

$$= (YA)^{XB} \bmod q$$

Example for Diffie-Hellman

- □ Alice and Bob agree on public values q and a, and determine their respective secrets XA and XB :
- □ Let $q = 5$ and $a = 3$
- □ Alice picks $XA = 2$, therefore $YA = a^{XA} \bmod 5 = 4$
- □ Bob picks $XB = 3$, therefore $YB = a^{XB} \bmod 5 = 2$
- □ Alice sends $YA = 4$ to Bob
- □ Bob sends $YB = 2$ to Alice
- □ Alice calculates: $K = (YB)^{XA} \bmod q = 2^2 \bmod 5 = 4$
- □ Bob calculates: $K = (YA)^{XB} \bmod q = 4^3 \bmod 5 = 4$

Ephemeral versus Static Diffie-Hellman Keys

- The generated DH keys can be either
  - static (to be reused)
  - ephemeral (only used once, e.g., for one session only)
- Ephemeral keys
  - provide forward secrecy, but no endpoint authenticity
    - Forward secrecy: If the current key is recovered by an adversary, it only effects the current session, but no past or future sessions
- Static keys
  - do not provide forward secrecy
  - do provide (implicit) endpoint authenticity
  - do not protect against replay-attacks

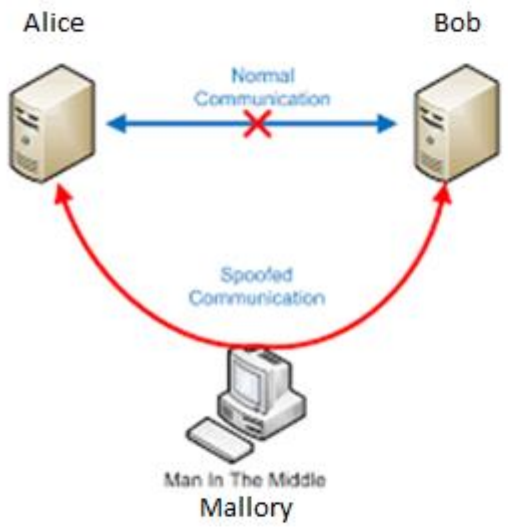
Example DH Parameters

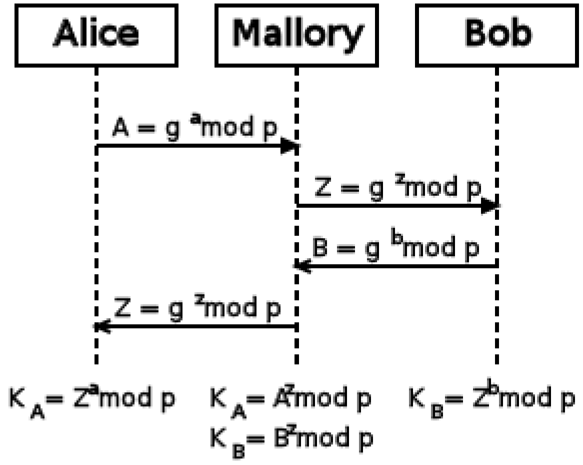
- Standardised, see https://www.ietf.org/rfc/rfc3526.txt
- Example 2048-bit MODP Group

[] == rounded

  - q = 2^2048 - 2^1984 - 1 + 2^64 * { [2^1918 pi] + 124476 }
  - q = FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1
    29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD
    EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
    E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED
    EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D
    C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F
    83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D
    670C354E 4ABC9804 F1746C08 CA18217C 32905E46 2E36CE3B
    E39E772C 180E8603 9B2783A2 EC07A28F B5C55DF0 6F4C52C9
    DE2BCBF6 95581718 3995497C EA956AE5 15D22618 98FA0510
    15728E5A 8AACAA68 FFFFFFFF FFFFFFFF
  - a = 2

DH and Man-in-the-Middle (MitM) Attacks





- Mallory is a MitM attacker with the ability to intercept, and fabricate messages
  - Not to confuse with a Meet-in-the-Middle attack (→ double-DES and triple-DES)
- Both Alice and Bob are unaware of Mallory's existence, as there is no mutual authentication and an unprotected communication link
- Alice and Bob exchange their shared values (A and B in the example), but these are intercepted by Mallory
- Mallory completes both key exchanges sending her own shared value Z to both Alice and Bob
- By doing so, Mallory establishes two individual (secure) connections with Alice and Bob
- Alice and Bob have no idea that they became victims of a MitM attack!

In-Class Activity: Diffie-Hellman MitM Attack

- Let q = 5 and a = 3;
- X_{Alice} = 2, therefore $Y_{Alice} = a^{XAlice} \bmod 5 = 4$
- X_{Bob} = 3, therefore $Y_{Bob} = a^{XBob} \bmod 5 = 2$
- X_{Malory} = 1, therefore $Y_{Malory} = a^{XMalory} \bmod 5 = 3$
- What session keys between
  - Alice and Malory
  - Malory and Bob

  are generated?
- Note: User A's key $K = (YB)^{XA} \bmod q$
- Note: User B's key $K = (YA)^{XB} \bmod q$

Solution

- Alice sends “4” to Bob, but this message is intercepted by Malory
- Bob sends “2” to Alice, but this message is intercepted by Malory
- Malory sends “3” to both parties, claiming to be either Bob or Alice
- Alice receives “3” and calculates K as follow: $K = 3^2 \bmod 5 = 4$
    - Malory calculates $4^1 \bmod 5 = 4$
- Bob receives “3” and calculates K as follow: $K = 3^3 \bmod 5 = 2$
    - Malory calculates $2^1 \bmod 5 = 2$
- Alice and Bob think they just mutually agreed on a shared secret key
- From this point onwards Malory as a MitM can read, manipulate and fabricate messages between Alice and Bob

The RSA Algorithm

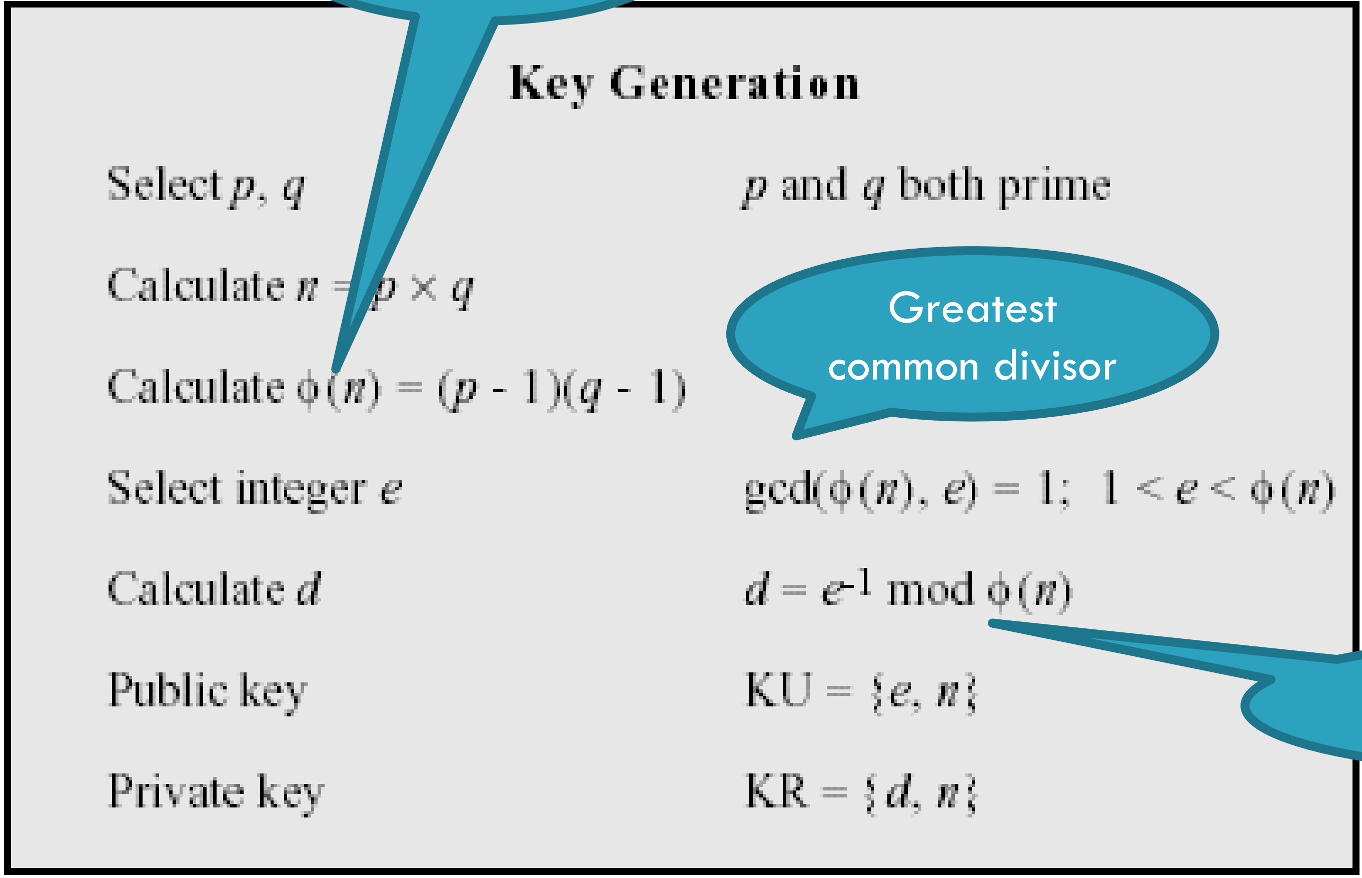
- Published by Rivest, Shamir and Adleman in 1977, but first discovered by Clifford Cocks (British mathematician and cryptographer) in 1973
- The RSA scheme works similar to a block cipher, where a plaintext M and a ciphertext C are integers between 0 and $n - 1$, i.e. elements of Z_n
- M can be a plaintext message (block), a hash value, or a private key picked by the sender to be shared with the message recipient
  - E.g., “ABC” = “01000001 01000010 01000011” = 4276803_{10}

- Principle: $C = M^e \bmod n$
  $M = C^d \bmod n = M^{ed} \bmod n$

- Public key $KU = \{e, n\}$
- Private key $KR = \{d, n\}$

- With n sufficiently large it is infeasible to determine d given e and n

Key Generation for the RSA Algorithm



Key Generation

Select p, q	p and q both prime
Calculate $n = p \times q$	
Calculate $\phi(n) = (p - 1)(q - 1)$	
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate d	$d = e^{-1} \bmod \phi(n)$
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

Euler's totient function Phi

Greatest common divisor

See next slide

Example

- Let $p = 7, \ q = 11$ and $n = pq = 77$
- $\phi(77) = (p - 1)(q - 1) = 6 \ x \ 10 = 60$
- Factorisation of $60 = 1 * 2 * 5 * 2 * 3$

  Therefore, the divisors of 60 are: $2, \ 3, \ 5$
- List of all integers x, $1<x<60$, with $GCD(60, x) = 1$:
  $7, \ 11, \ 13, \ 17, \ 19, \ 23, \ 29, \ 31, \ 37, \ 47, \ 49, \ 53, \ 59$
-

  Note that these integers either
  - are prime numbers (that cannot share a common divisor with 60), or
  - do not share a common divisor with 60 (i.e., 7 and 49)

Example (continued)

- Let $e = 7$
- Choose d with $ed = 1 \bmod \phi(pq) \Leftrightarrow$

  $7d = 1 \bmod 60 \Leftrightarrow \mathbf{7d \bmod 60 = 1}$

  $7*1 \bmod 60 = 7$ $\quad$ $7*2 \bmod 60 = 14$ $\quad$ $7*3 \bmod 60 = 21$

  $7*4 \bmod 60 = 28$ $\quad$ $7*5 \bmod 60 = 35$ $\quad$ $7*6 \bmod 60 = 42$

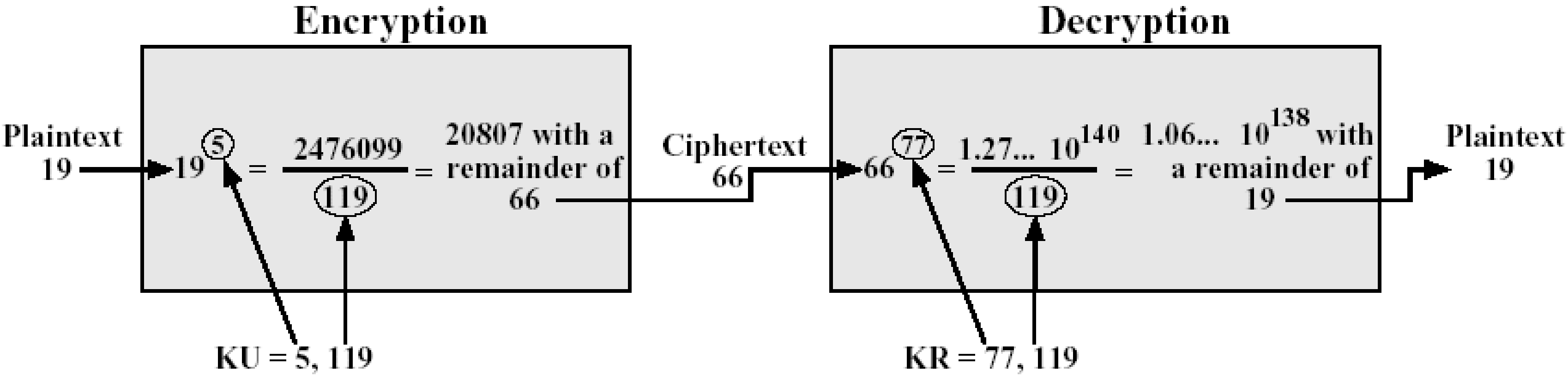
  $7*7 \bmod 60 = 49$ $\quad$ $7*8 \bmod 60 = 56$ $\quad$ $7*9 \bmod 60 = 3$

  $7*10 \bmod 10 = 28$ $\quad$ $7*11 \bmod 60 = 17$ $\quad$ $7*12 \bmod 60 = 24$

  … $\quad$ $\mathbf{7*43 \bmod 60 = 1}$

- Therefore $d = 43$
- Therefore $KU = (7, 77)$ and $KR = (43, 77)$
- Note there are better / more efficient algorithms (i.e. the Extended Euclidean Algorithm) to calculate d

Example for an Encryption/Decryption



Encryption

Plaintext 19

$$19^{5} = \frac{2476099}{119} = 20807 \text{ with a remainder of } 66$$

KU = 5, 119

Ciphertext 66

Decryption

$$66^{77} = \frac{1.27\ldots\ 10^{140}}{119} = 1.06\ldots\ 10^{138} \text{ with a remainder of } 19$$

KR = 77, 119

Plaintext 19

- Obvious drawbacks:
  - Very large numbers are to be computed
    - Ordinary integer or floating-point variables don’t work
    - Instead, large number libraries need to be used
  - This makes RSA encryption / decryption is very slow!

Computational Aspects of Public Key Cryptography

- Assume you have to evaluate the expression $C = 503^{23} \bmod 899$ as part of the encoding process
  - Note that the modulus is small enough to fit into an integer variable

- $503^{23} = 1.36792931379540842325043971010 6 \text{ x } 10^{62}$ cannot be properly represented using an ordinary integer or floating-point variable!

- In order to solve this problem, the exponentiation must be broken down into smaller steps, e.g.

  - $503^{23} \bmod 899 = ((503^{6} \bmod 899) \text{ x } (503^{6} \bmod 899)$
    $\text{x } (503^{6} \bmod 899) \text{ x } (503^{5} \bmod 899)) \bmod 899$

  - $503^{6} \bmod 899 = ((503^{3} \bmod 899) \text{ x } (503^{3} \bmod 899)) \bmod 899$
  - $503^{5} \bmod 899 = ((503^{3} \bmod 899) \text{ x } (503^{2} \bmod 899)) \bmod 899$
  - $503^{3} \bmod 899 = ((503^{2} \bmod 899) \text{ x } 503) \bmod 899$

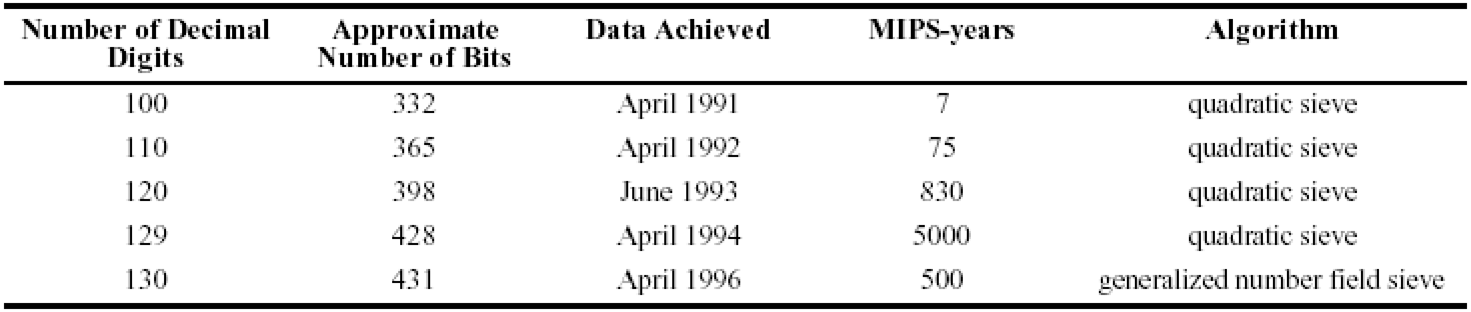
Computational Aspects of Public Key Cryptography

- … or even iteratively:
  $503^{23} \text{ mod } 899 =$
  $((((((503^{2} \text{ mod } 899) \text{ x } 503) \text{ mod } 899) \text{ x } 503) \text{ mod } 899) \text{ x } \cdots \text{ x } 503) \text{ mod } 899$

- This expression consists of 22 nested multiplications and 22 nested modulus operations and can be easily calculated by using a loop
- However, once a single number squared is too large to fit into a 32-bit or 64-bit (unsigned) integer variable, a big number library must be used

The Security of RSA

- There are various angles to attack the RSA algorithm:
  - Brute force: Trying all possible private keys (not a great idea!)
  - Mathematical attacks: Factor n (which is the product of two primes); see some very old data below:



Number of Decimal Digits	Approximate Number of Bits	Data Achieved	MIPS-years	Algorithm
100	332	April 1991	7	quadratic sieve
110	365	April 1992	75	quadratic sieve
120	398	June 1993	830	quadratic sieve
129	428	April 1994	5000	quadratic sieve
130	431	April 1996	500	generalized number field sieve

  - See also (for some more recent data) https://en.wikipedia.org/wiki/RSA_numbers#RSA-704
  - Timing attacks: Based on analysis of the run time of an decryption algorithm

Breaking RSA

- Consider the key pair (e, n) and (d, n) or simply (e, n) and d
  - $n = p * q$, with p and q being large (secret!) primes
- Factorising n is unfeasible for very large n
- However, let's assume n can be factored into p and q
- The adversary can now do the following calculations:
  - $\phi(n) = (p - 1) * (q - 1)$
  - Identify d, so that $e * d = 1 \bmod \phi(n)$
    - e is known, use the aforementioned Extended Euclidean Algorithm

Step 1: Factorise N

```
// This is a very lightweight integer factoring algorithm, not very efficient or
// sophisticated.
// Assume n is the product of two primes p1 and p2
void factorise(int n) {
        int i;
        for (p1 = 2; i <= sqrt(n); i++) {
                if (n % p1 == 0)
                        printf(“n = %d; p1 = %d; p2 = %d\n”), n, p1, n / p1);
                break;
}
// Note that the integer values above would be replaced with large number
// representations, i.e., BIGNUM in OpenSSL
```

Step 2: Determine e

```
// We know p and q (n was successfully factorised), d is in the public key KR= d, n
// This is again a very lightweight algorithm, not very efficient or sophisticated.
int breakRSA(int p, int q, int d) {
        int prod, found = 0, start = 1, df = -1;
        int phi = (p -1) * (q – 1);
        while ((!found) && (start < phi)) { // exit if needed
                prod = d * start;
                if (prod % phi == 1) found = 1;
                else start++;
        }
        if (found) df = start;
        return (df);
}
// Note that the integer values above would be replaced with large number
// representations, i.e., BIGNUM in OpenSSL
```

How to choose p and q

- When choosing p and q, the following should be considered:
  1. p <> q, as p = q = sqrt(n)
  2. Neither p or q must not be “small”, as factorising could produce a result in a reasonable amount of time (see previous slide “Step 1: Factorise N”)
  3. p must not be similar in size to q, because of *Fermat's method of factoring a composite number N*:
     - N can be represented as the difference of two squares:
       - $p * q = \underline{N \Leftrightarrow a^2 - b^2} \Leftrightarrow (a - b)(a + b)$ [== $p * q$]
     - $N = a^2 - b^2$ can be rewritten as: $b^2 = a^2 - N$
     - To find a solution, iterate through a (starting with round(sqrt(N))), until $a^2 - N$ is a square number (i.e. b^2)

Fermat’s Factoring Algorithm

```
// This function assumes N can be factorised. It returns N’s factors
// p and q, using “pass by reference” pointers, so that both values
// are returned.
void fermatFactor(int N, int *p, int *q) {
    int a = ceiling(sqrt(N)); // start value for a
    int b2 = a * a - N; // see last slide
    while (sqrt(b2) * sqrt(b2) <> b2) { // is b2 a square?
        a = a + 1; // No, so increment a …
        b2 = a * a – N; // … and update b2
    }
    *p = a - sqrt(b2);
    *q = a + sqrt(b2);
}
```

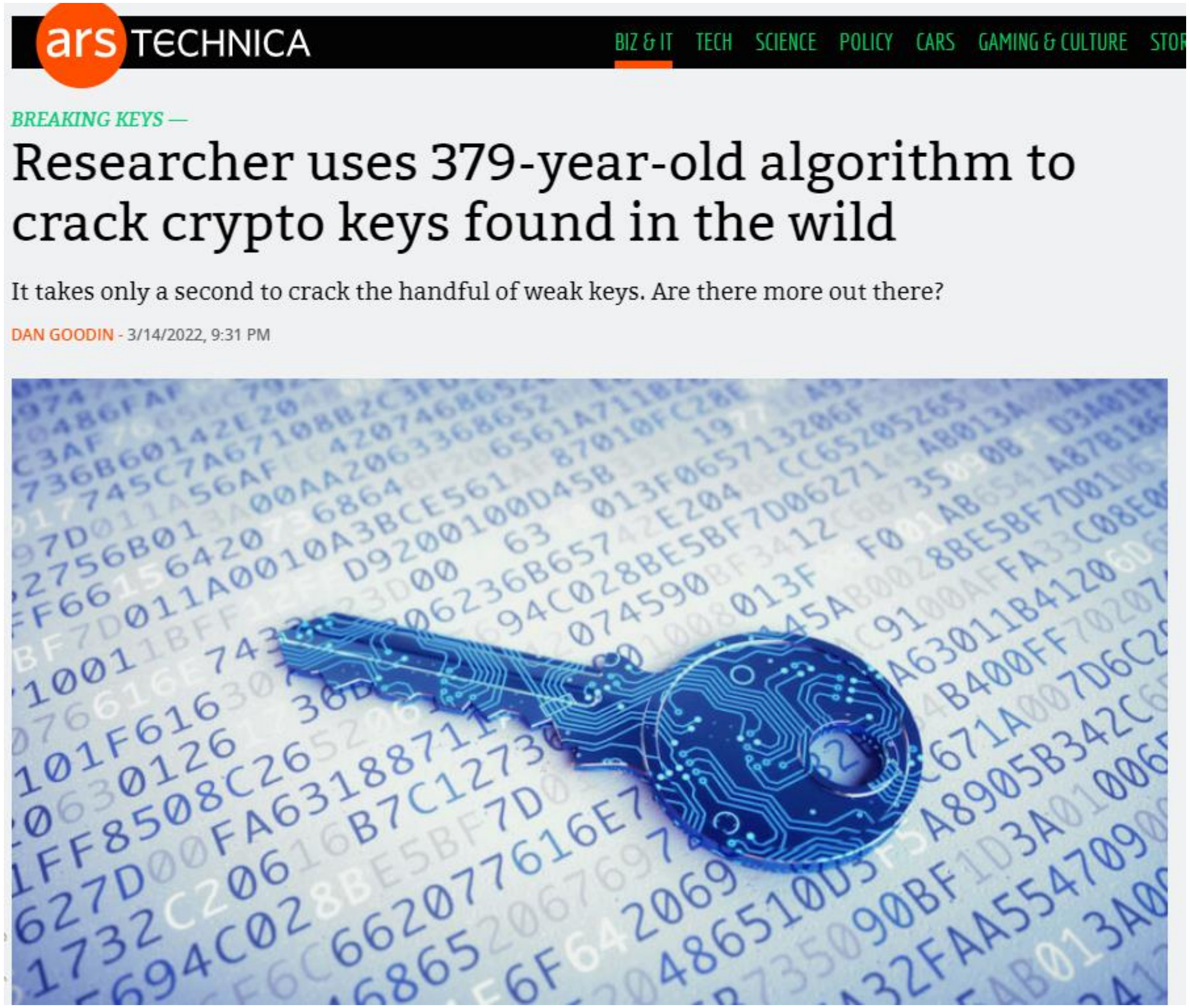
If p (= a - b) and q (= a + b) are similar in size, it takes only a small number of iterations over a to find a solution

Example

1. n = 33 (based on secret values p = 3 and q = 11)
2. First iteration: a = 6 (i.e., ceiling(sqrt(33)):
    1. b2 = 6 * 6 – 33 = 3
    2. b2 is not a square number
    3. a = a + 1
3. Second iteration: a = 7:
    1. b2 = 7 * 7 – 33 = 16
    2. b2 is a square number
4. Calculate p and q:
    1. p = 7 - sqrt(16) = 3
    2. q = 7 + sqrt(16) = 11

Breaking RSA in Practise

https://arstechnica.com/information-technology/2022/03/researcher-uses-600-year-old-algorithm-to-crack-crypto-keys-found-in-the-wild/



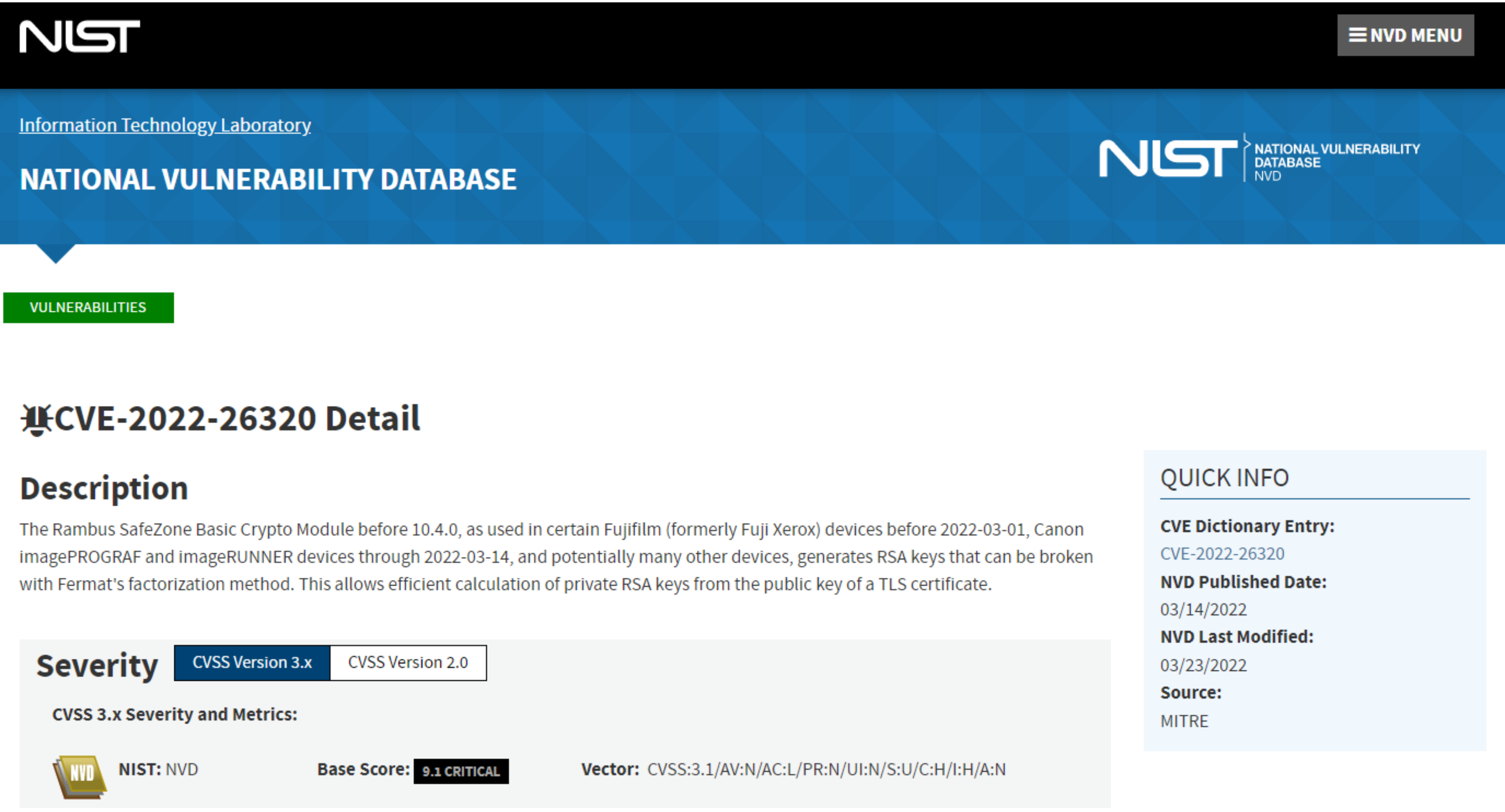
BREAKING KEYS —

Researcher uses 379-year-old algorithm to crack crypto keys found in the wild

It takes only a second to crack the handful of weak keys. Are there more out there?

DAN GOODIN - 3/14/2022, 9:31 PM

CVE-2022-26320



Information Technology Laboratory

NATIONAL VULNERABILITY DATABASE

CVE-2022-26320 Detail

Description

The Rambus SafeZone Basic Crypto Module before 10.4.0, as used in certain Fujifilm (formerly Fuji Xerox) devices before 2022-03-01, Canon imagePROGRAF and imageRUNNER devices through 2022-03-14, and potentially many other devices, generates RSA keys that can be broken with Fermat's factorization method. This allows efficient calculation of private RSA keys from the public key of a TLS certificate.

Severity

CVSS 3.x Severity and Metrics:

NIST: NVD **Base Score:** 9.1 CRITICAL **Vector:** CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N

QUICK INFO

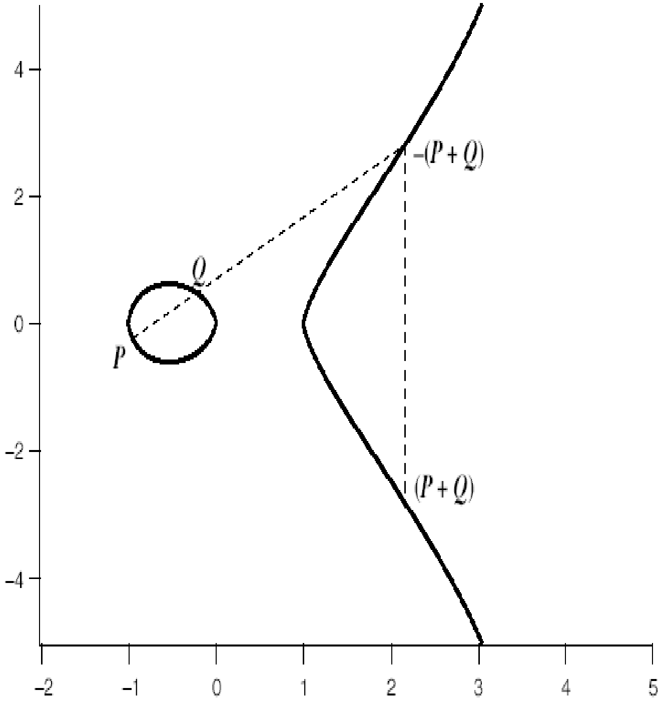
CVE Dictionary Entry:
CVE-2022-26320
NVD Published Date:
03/14/2022
NVD Last Modified:
03/23/2022
Source:
MITRE

Elliptic Curve Cryptography (ECC)

- □ Traditional methods exploit the properties of arithmetic using large finite groups Z_n with n having a typical size of 1024 bits, i.e. 309 decimal digits
- □ The security depends on the difficulty of factorising large numbers or calculating discrete logarithms
- □ Using large numbers makes such algorithms computationally expensive
- □ In ECC, Z_n is replaced by points of an elliptic curve, making the discrete log calculation problem different and much harder compared to the discrete log in ordinary groups

Elliptic Curve Groups

- Elliptic curves are based on simplified cubic equations, e.g.
  $y^2 = x^3 + ax + b$
  where a and b are real numbers
- The curve shown here is defined by the equation
  $y^2 = x^3 - x$ (i.e., a = -1 and b = 0)
- To plot such a curve, we need to compute
  $y = \mathrm{sqrt}(x^3 + ax + b)$
- Since the shape of the curve depends on a and b, ECs can be described as $E(a, b)$
  - The above curve can be written as $E(-1, 0)$
- In order to operate on elliptic curves, we need to introduce an operation that is equivalent to the addition as well as a “0” element

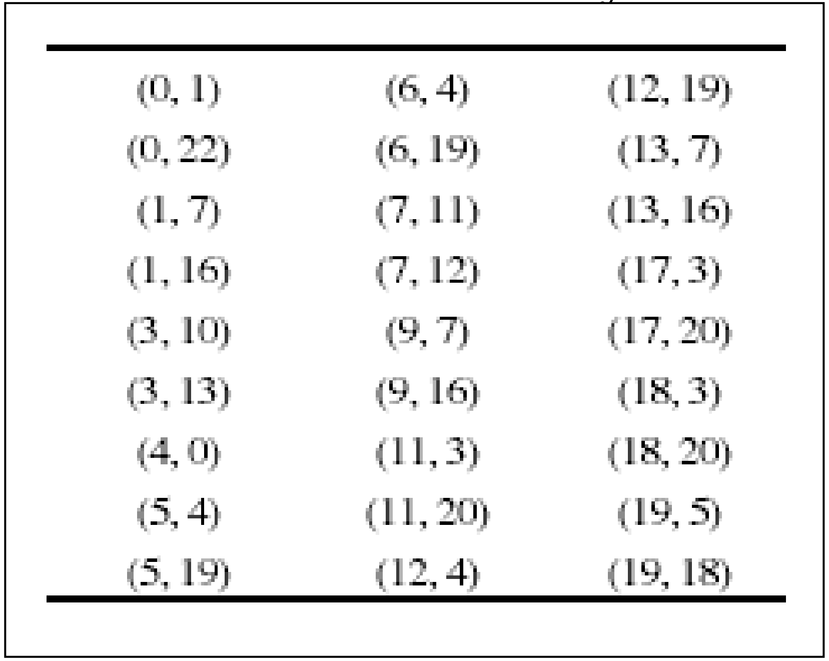


Elliptic Curves over a Finite Field

- In order to have values (x, y) within Z_p, the modulus operation is used again:

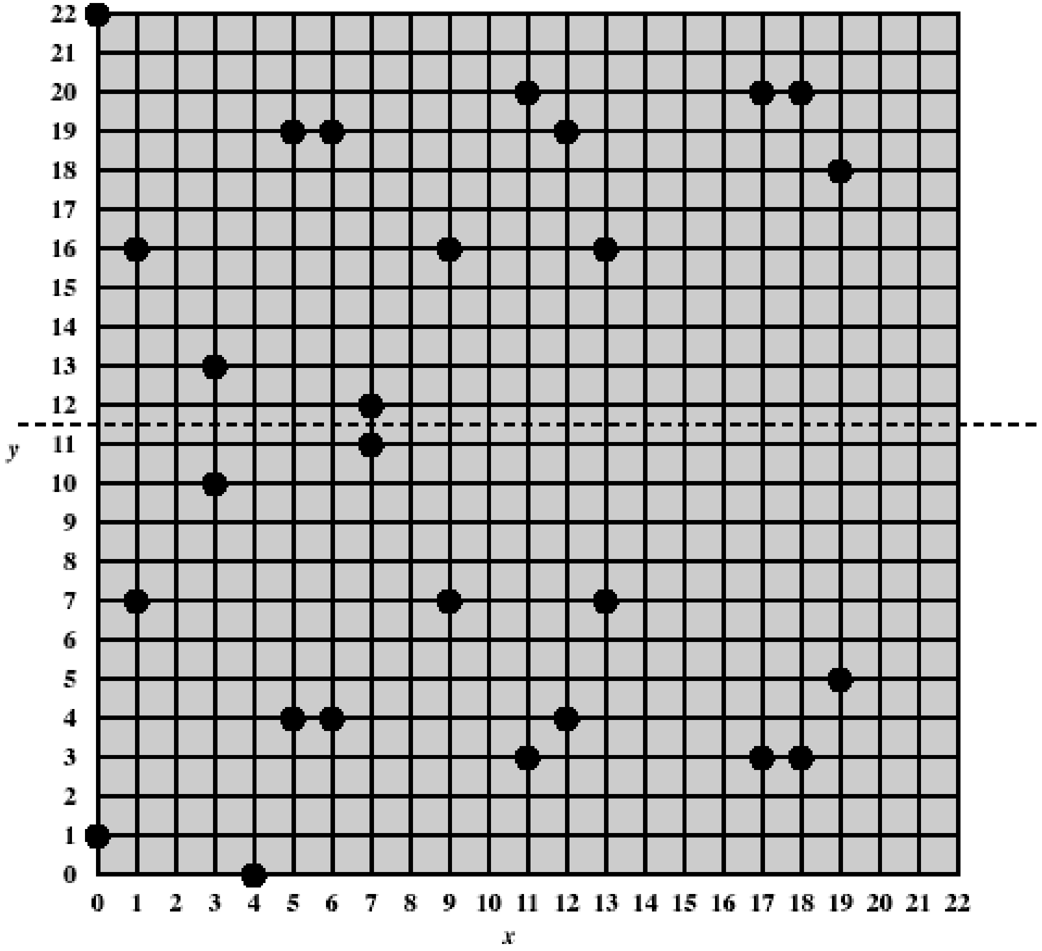
  $$y^2 \bmod p = (x^3 + ax + b) \bmod p$$

- p is either a prime number or $p = 2^m$
- We only consider pairs (x, y), where both x and y are integer values
- Example: Table of all integer solutions for $E_{23}(1,1)$



(0, 1)	(6, 4)	(12, 19)
(0, 22)	(6, 19)	(13, 7)
(1, 7)	(7, 11)	(13, 16)
(1, 16)	(7, 12)	(17, 3)
(3, 10)	(9, 7)	(17, 20)
(3, 13)	(9, 16)	(18, 3)
(4, 0)	(11, 3)	(18, 20)
(5, 4)	(11, 20)	(19, 5)
(5, 19)	(12, 4)	(19, 18)

The Elliptic Curve $E_{23}(1,1)$

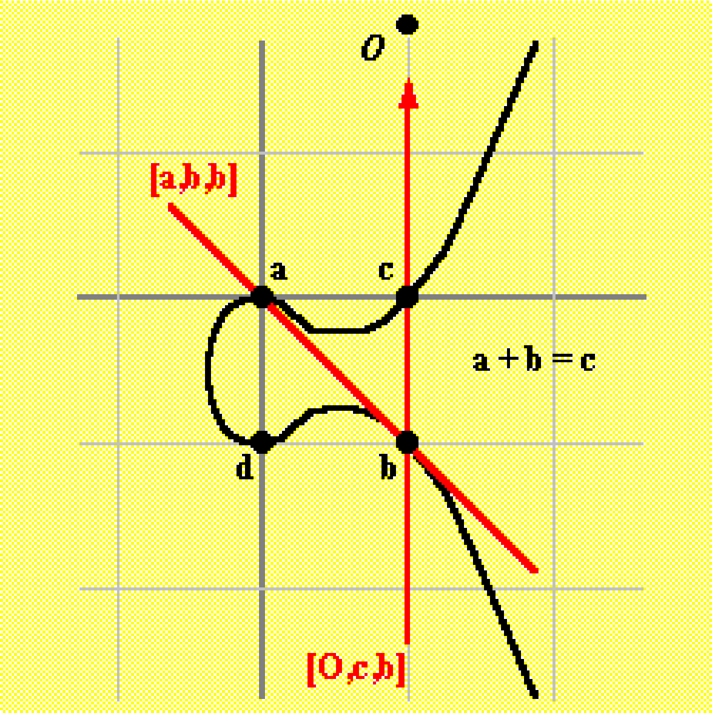


Adding Points on an Elliptic Curve

- ECC requires the equivalent of an addition on $E_p(A,B)$ of two points a and b
- This is done (geometrically) as follows:
  - Draw a straight line through a and b to find the third intersecting point w,
  - then draw a vertical line through w to find the intersecting point c (that's the sum)
- Every line intersects the curve three times (tangents are counted twice), e.g., the line through a and b intersects a "third" point b. We name this line [a,b,b]
- O is called the origin, or point at infinity
- We can say

  a + b = c　　　　a + d = b + c = O

  a + a = b　　　　a + O = a



ECC over a Finite Field: Addition

- There's p as defined before
- Addition of two field elements $S = (x_S, y_S)$ and $Q = (x_Q, y_Q)$ with $S <> -Q$:
  - $S + Q = R = (x_R, y_R)$
  - $x_R = (L^2 - x_S - x_Q) \bmod p$
  - $y_R = (L (x_S - x_R) - y_S) \bmod p$
  - L is either
    - $((y_Q - y_S) / (x_Q - x_S)) \bmod p$, if $S <> Q$, or
    - $((3 x^2{}_S + a) / (2 y_S)) \bmod p$, if $S = Q$

ECC over a Finite Field: Addition and Multiplication

- The addition of two elliptic points P and Q consists of a number of integer operations (mod q):
  - 5 or 6 subtractions
  - 1 or 4 multiplications
  - 1 division
- A multiplication $(\mathrm{P} * \mathrm{Q})$ is done via consecutive additions
- A scalar multiplication $(\mathrm{x} * \mathrm{Q})$ with some scalar x is the operation of successively adding a point Q along an elliptic curve to itself x times (i.e. $\mathrm{Q} + \mathrm{Q} + \mathrm{Q} + \cdots + \mathrm{Q})$

ECC Diffie-Hellman

- Similar to conventional Diffie-Hellman, but operates of finite EC field:
  - Users A & B select a suitable curve $E_p(a, b)$
  - Users select base point (equivalent to primitive root) $G=(x_1, y_1)$
  - User A & B select private keys n_a and n_b
  - Users A & B compute public keys PA and PB
  - Shared keys are exchanged
  - Secret key K is computed

ECC Diffie-Hellman Example

- Use $E_{211}(0, -4)$ that is equivalent to y^2 mod 211 = $(x^3 - 4)$ mod 211

- Choose G = (2, 2)

- User A chooses n_a = 121, so A's public key PA is:
  121 * G = 121 * (2, 2) = (115, 48)

- User B chooses n_b = 203, so B's public key PB is:
  203 * G = 203 * (2, 2) = (130, 203)

- The shared secret key K is 121 * (130, 203) = 203 * (115, 48) = (169, 69)

- Note:
  - ECC-DH (or ECDH for short) can be compromised via a MitM!
  - We still use a BIGNUM integer representation, but the range of values is significantly smaller, and operations can be executed much quicker (see next slide)

Comparable Key Sizes for Equivalent Security

Symmetric scheme (key size in bits)	**ECC-based scheme (size of *p* in bits)**	**RSA (modulus size in bits)**
56	112	512
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360

FYI: Curve25519

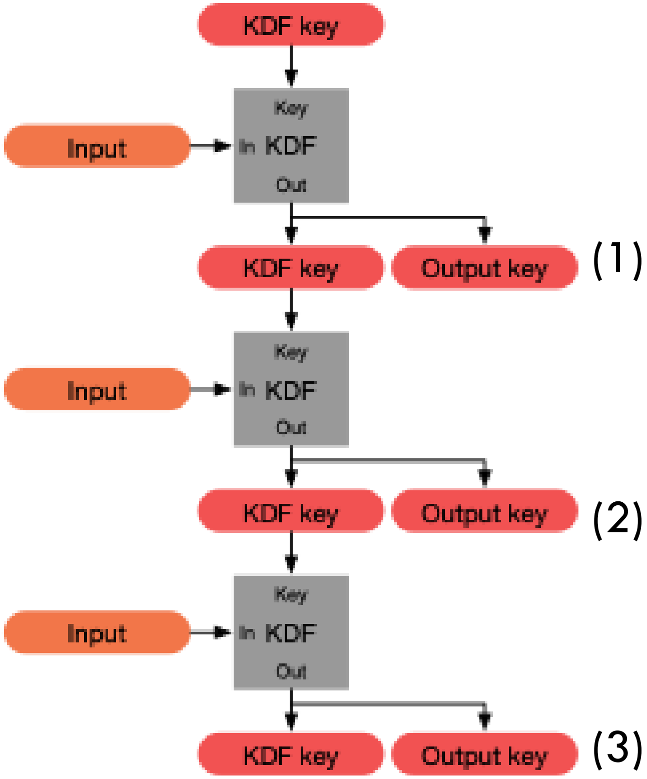
- Curve25519 is an elliptic curve offering 128 bits of security (with 256 bits key size) and designed for use with the elliptic curve Diffie–Hellman (ECDH) key agreement scheme
- It is one of the fastest ECC curves and is not covered by any known patents
- It was first released by the cryptologist Daniel J. Bernstein in 2005
- In 2013, interest began to increase considerably when it was discovered that the NSA had potentially implemented a backdoor into the most common EC encryption method
    - i.e. the P-256 curve based Dual_EC_DRBG algorithm
- Today it is the de facto alternative to P-256
- Its reference implementation is public domain software

The Double-Ratchet Algorithm[1]

- The **Double Ratchet algorithm** is a cryptographic protocol used by two parties to exchange encrypted messages
  - Messages are encrypted using (fast) symmetric key algorithms (e.g., AES)
  - Every message that is exchanged in either direction is encrypted using a different private key
- The algorithm is implemented in the Signal protocol, which in turn is used in secure messaging apps such as the Signal app and WhatsApp
- It ensures forward secrecy and post-compromise security, making conversations secure even if previous keys are compromised
- (Perfect) forward secrecy and post-compromise security are properties of secure communication protocols
  - **Forward security** ensures the confidentiality of past sessions even if long-term keys are compromised
  - **Post-compromise security** ensures the security of future communications even after an initial compromise

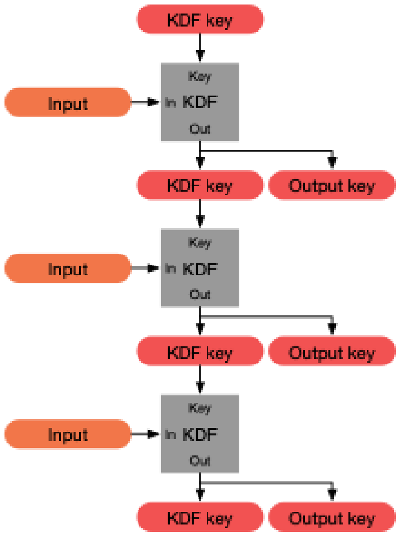
Key Derivation Function (KDF) and KDF Chains

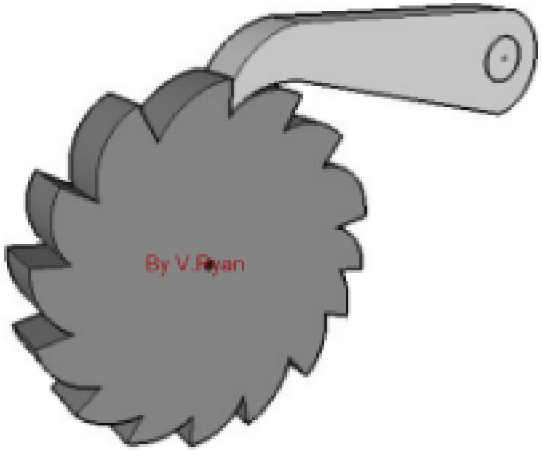
- A **KDF** is a cryptographic function that
  - is used to create a new secret key for each message
  - takes a secret (KDF key) and some (Input) data, and returns an output
  - looks like a "one-way" function (i.e., a hash function)
- In a **KDF chain** some of the KDF output is used as an (Output key) and some is used to make a new (KDF key)
- If two endpoints agree on the same initial (KDF key) and the same (Input), they create the same sequence of output keys, and can exchange messages securely
- A KDF chain guarantees forward security, but not automatically post-compromise security
  - Consider output key (2) being recovered by an attacker:
  - The attacker cannot calculate key (1)
  - The attacker is only prevented from calculating Output key (3), if Input is a secret shared by both endpoints



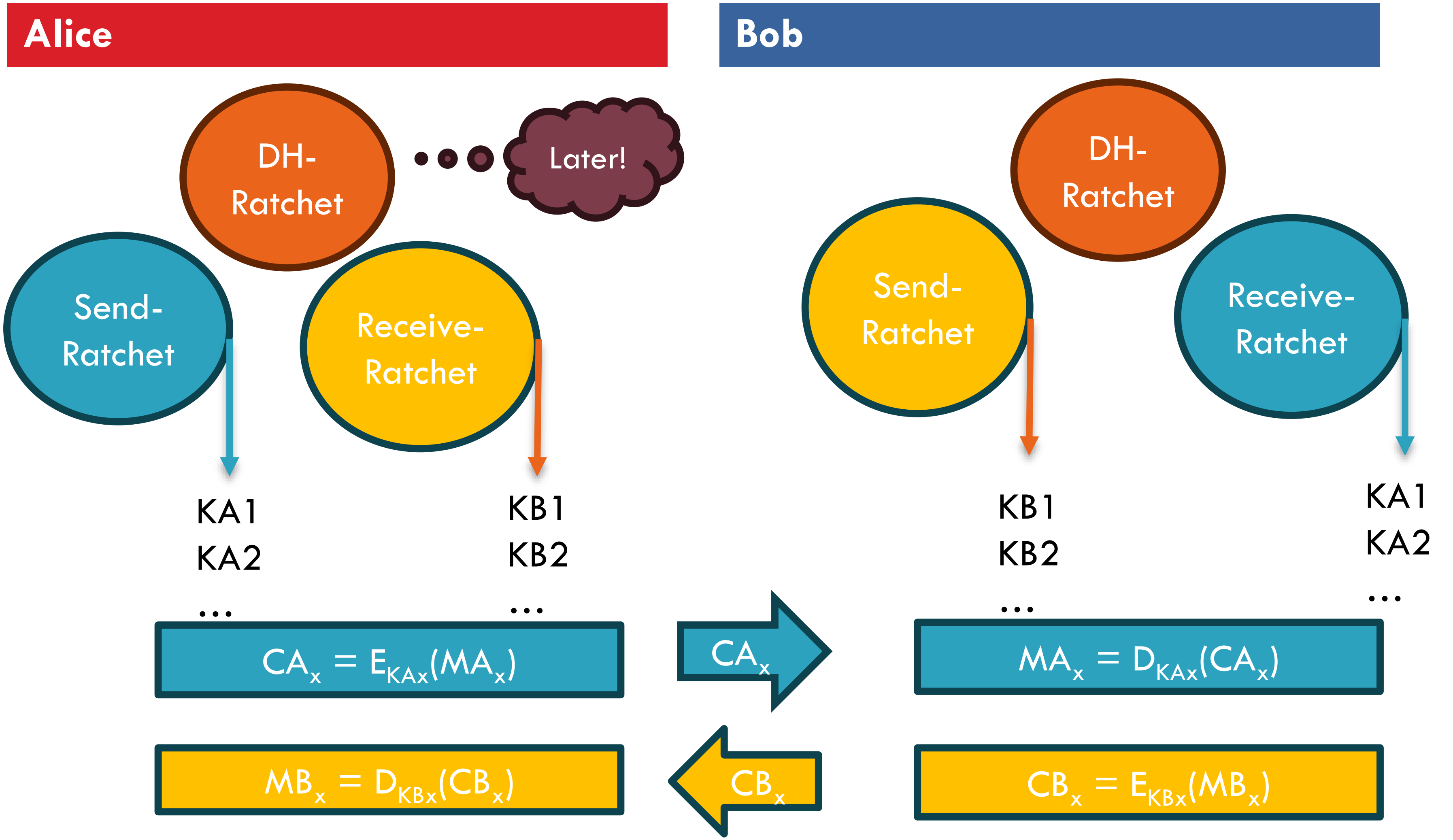
KDF Chains

- A KDF chain is like a ratchet, which only goes in one direction
  - each step provides a different output (KDF key|| Output key)
- Both Alice and Bob have both a “send” and “receive” ratchet each
- Alice’s “send” and Bob’s “receive” ratchet are initialised using the same initial KDF and the same *Input* key (and visa versa)
- Every time a message is to be sent by either side, it is encrypted first using a new encryption key (*Output key*) that is generated by invoking the KDF (i.e., the “sender” ratchet)
- Similarly, every time the receiver receives a new message it calculates the (same) key for message decryption by invoking the KDF (i.e., the “receiver” ratchet)





Sender and Receiver Ratchet



Alice

DH-Ratchet

Later!

Send-Ratchet

Receive-Ratchet

KA1
KA2
…

KB1
KB2
…

$CA_x = E_{KAx}(MA_x)$

$MB_x = D_{KBx}(CB_x)$

CA_x

CB_x

Bob

DH-Ratchet

Send-Ratchet

Receive-Ratchet

KB1
KB2
…

KA1
KA2
…

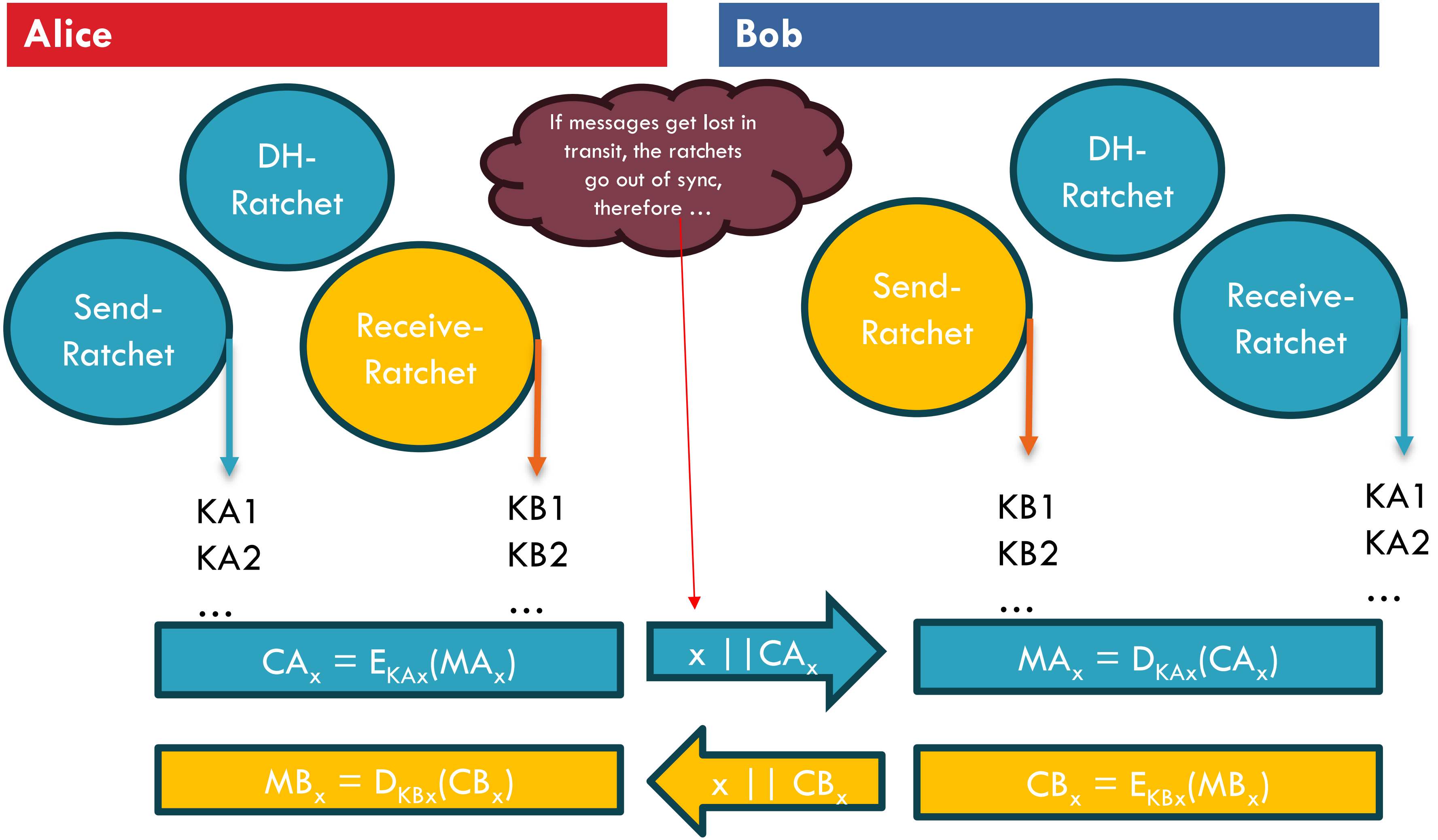
$MA_x = D_{KAx}(CA_x)$

$CB_x = E_{KBx}(MB_x)$

Explanations

- $K\{A|B\}_x$ is a secret key used by A or B for encoding and decoding a message (e.g., KA_5 or KB_7)
  - x is simply an incremented index value (i.e., 1, 2, 3,…)
- $M\{A|B\}_x$ are (indexed) plaintext messages generated by A or B (e.g., MA_5 or MB_7)
- $C\{A|B\}_x$ is the corresponding ciphertext
  - E.g., MA_3 <-> CA_3
- E() and D() are corresponding encryption and decryption functions that use a key KAx (e.g., $D_{KA5}(CA_5)$)

Synchronising Sender and Receiver Ratchets to compensate for lost Messages



Alice

Bob

If messages get lost in transit, the ratchets go out of sync, therefore …

DH-Ratchet

Send-Ratchet

Receive-Ratchet

KA1
KA2
…

KB1
KB2
…

$CA_x = E_{KAx}(MA_x)$

$x \mid\mid CA_x$

$MA_x = D_{KAx}(CA_x)$

$MB_x = D_{KBx}(CB_x)$

$x \mid\mid CB_x$

$CB_x = E_{KBx}(MB_x)$

DH-Ratchet

Send-Ratchet

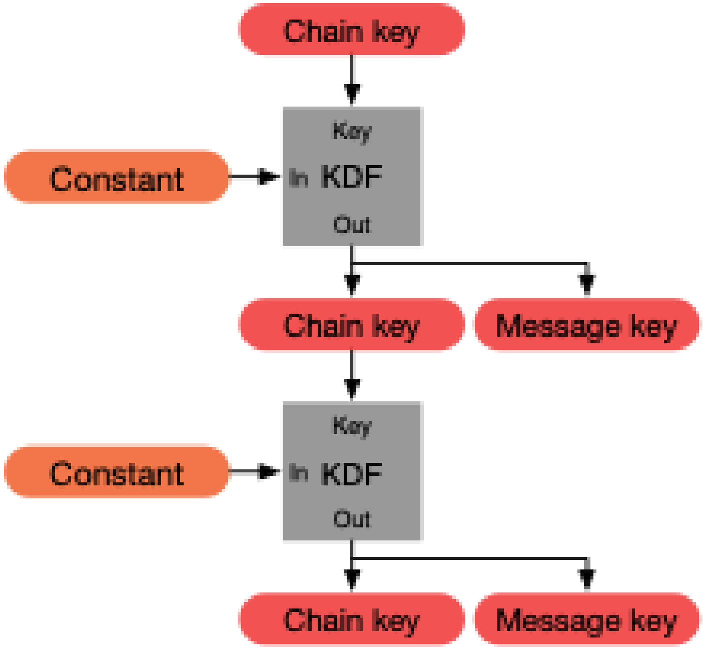
Receive-Ratchet

KB1
KB2
…

KA1
KA2
…

Symmetric Key Ratchet

- "Send" and "receive" ratchets are also called the **symmetric-key ratchets**
- Since every message sent is encrypted with a unique *Message key* (see diagram), the receiver may have to buffer generated (decryption) keys to deal with packets received out-of-order
- Here KDF keys are called *(Chain keys)*
- *The sequence of* generated *chain keys is called* a **sending chain** / **receiving chain**
- Here KDF chains use a (secret) *(Constant)* as a 2nd input to provide post-compromise security



The Diffie-Hellman Ratchet

- As Alice and Bob exchange messages, they also exchange new Diffie-Hellman public keys to generate shared secret keys
- These secret keys become the *input* to another KDF chain, the **root chain**
  - This is called the **Diffie-Hellman ratchet**
- The output keys from the root chain provide for new KDF chain keys for the sending and receiving ratchet
- The complete construct is called a Double Ratchet, consisting of the symmetric key ratchets and the DH ratchet, which require KDF keys for three chains:
  - a **sending chain and a receiving chain** (linked to the "send" and "receive "ratchets)
    - With Alice's sending chain matches Bob's receiving chain, and vice versa
  - a **root chain** (linked to the DH-ratchet)

The Diffie-Hellman Ratchet

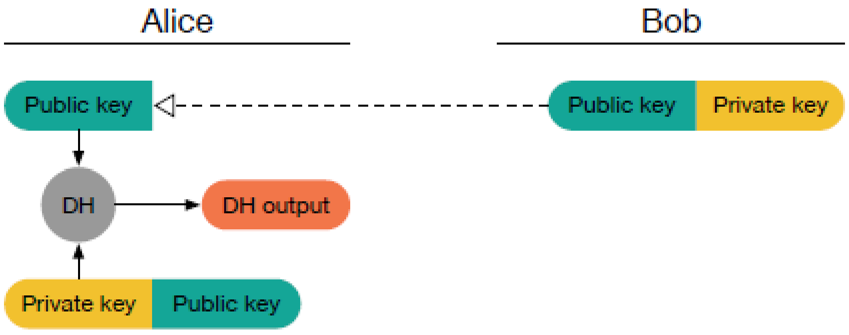
- To implement the DH ratchet, each party generates a DH key pair (a Diffie-Hellman public key and private key) which becomes their current ratchet key pair
- Every message from either party begins with a header which contains the sender’s current DH-ratchet public key
- When a new ratchet public key is received from the other party, a DH ratchet step is performed which replaces the local party’s current ratchet key pair with a new key pair
- This results in a “ping-pong” behavior as the parties take turns replacing ratchet key pairs

Stepping through the DH-Ratchet: Step 1

- Alice receives Bob’s ratchet’s public key
  - Alice’s ratchet’s public key isn’t yet known to Bob
- As part of the initialisation Alice performs a DH calculation using her ratchet’s (Private key) and Bob’s ratchet’s (Public key)

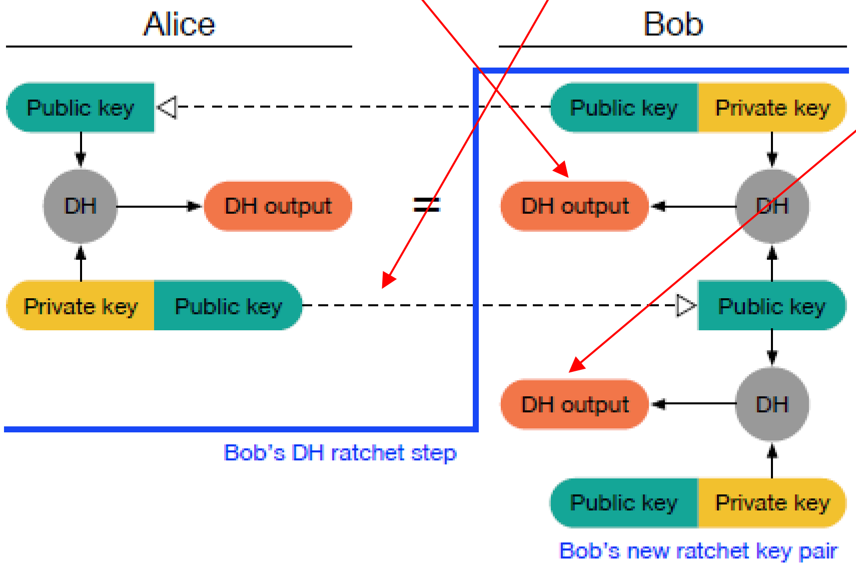
Bob’s rachet’s public key

Alice’s rachet’s private and public key



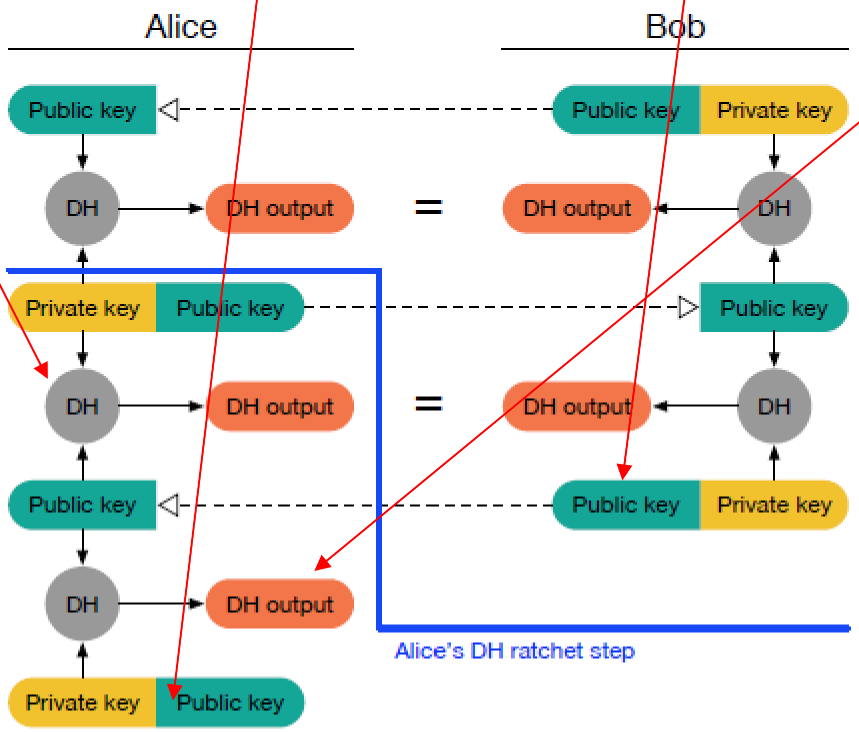
Stepping through the DH-Ratchet: Step 2

- Alice’s initial messages advertise her ratchet’s public key
- Once Bob receives one of these messages, he performs a DH ratchet step (consisting of two (DH) steps, i.e., Diffie-Hellman key exchange calculations):
  - He calculates the **DH output** between Alice’s ratchet’s public key and his ratchet’s (Private key), which equals Alice’s initial (DH output)
  - Bob then calculates a **new ratchet key pair** and calculates a new **DH output:**



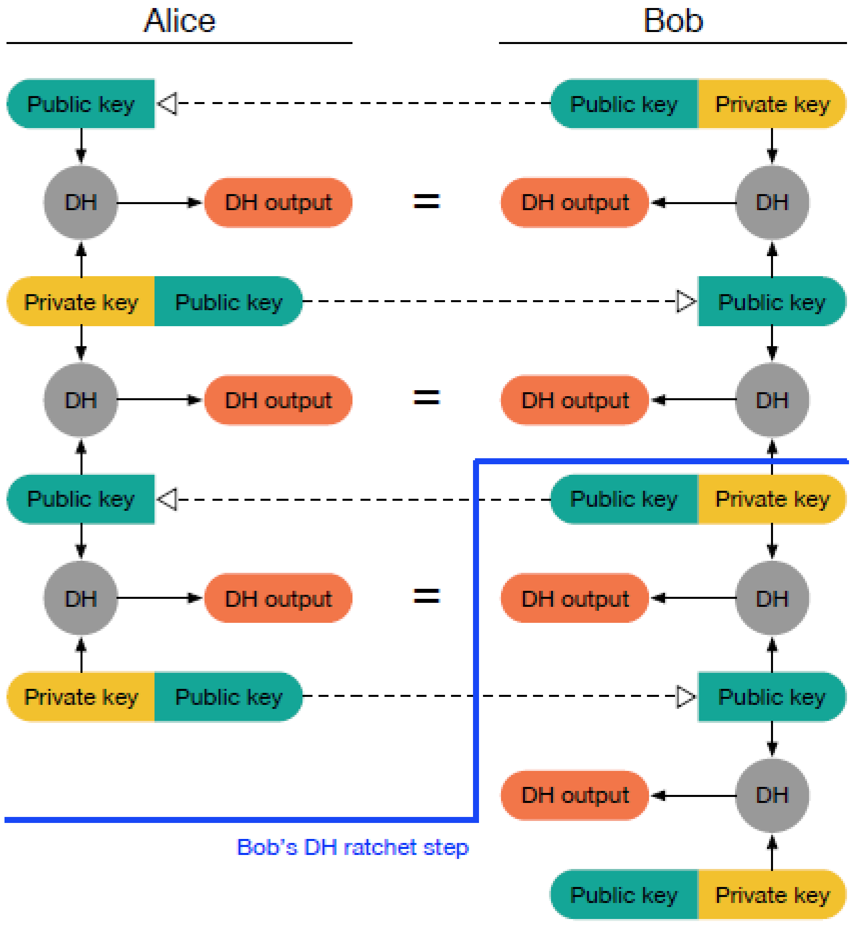
Stepping through the DH-Ratchet: Step 3

- Messages sent by Bob advertising his new Public key are received by Alice, who does a similar step comprising:
  - A (DH) operation using her current Private key and bob's new Public key will result in a DH output identical to the one calculated by Bob
  - She creates a new Private / Public key and calculates a new DH output :



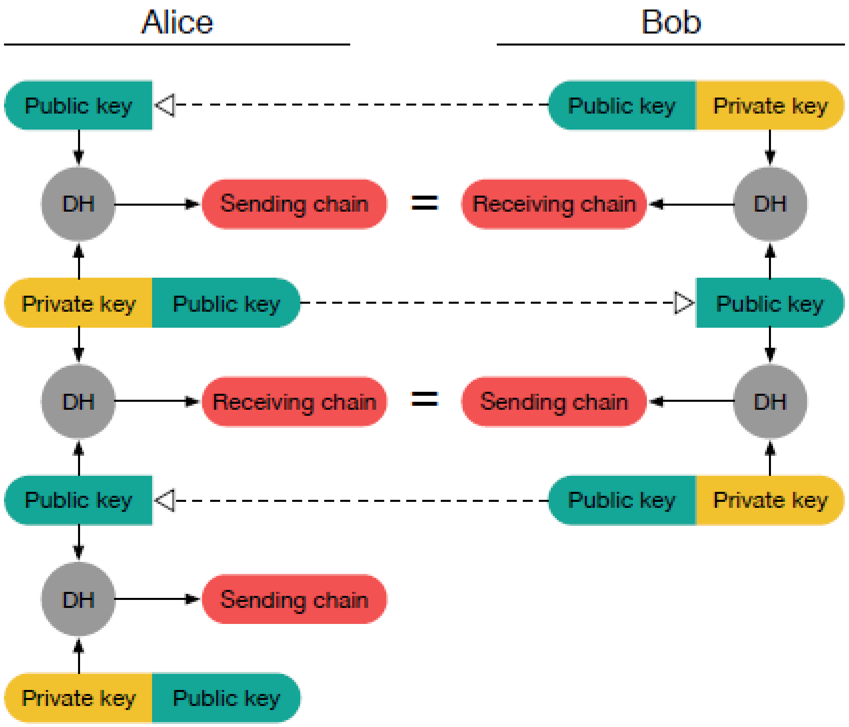
Stepping through the Diffie-Hellman Ratchet: Step 4+

- Messages sent by Alice advertise her new public key
- Bob receives one of these messages and perform a second DH ratchet step, and so on



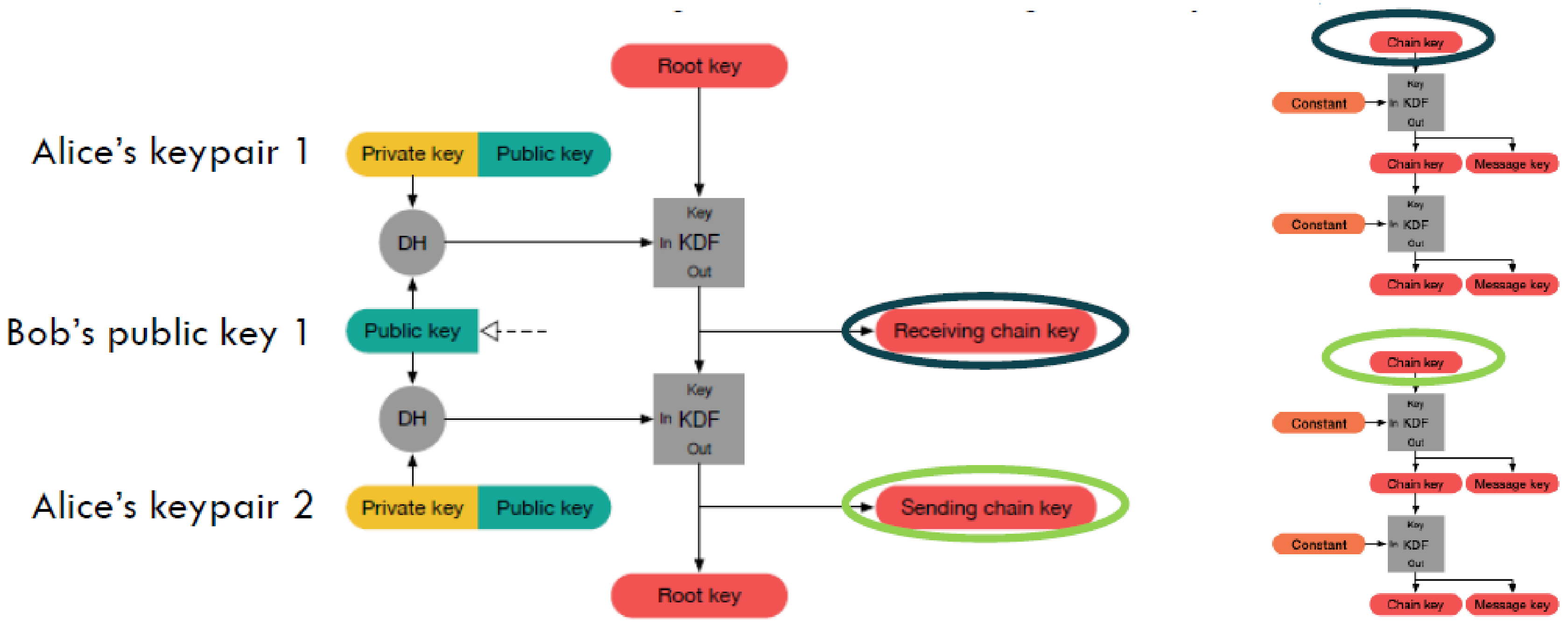
Deriving Sending and Receiving Chains Keys

- The **DH outputs** generated during each DH ratchet step are used to derive new sending and receiving chain keys for Alice's and Bob's symmetric key ratchets
- The DH outputs are not used directly, but go through a DH ratchet first (see next slide)



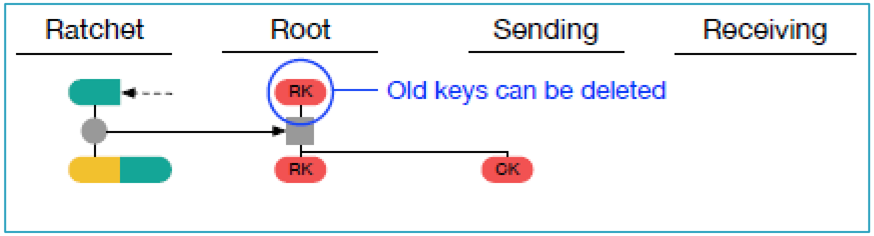
Deriving Sending and Receiving Chains

- This diagram shows the complete process from Alice’s perspective:
  - The **Root Key** is a shared secret with Bob, determined via (ECC-) DH at the beginning of the protocol / session
  - The DH output (as calculated in previous slides), together with the Root key, is processed by the DH ratchet in the centre of the diagram to create a *Receiving chain key*
  - Bob’s public key, together with Alice’s *Private key* of her 2nd generated keypair is used for another KDF invocation that generates the *Sending chain key* and a new *Root key*

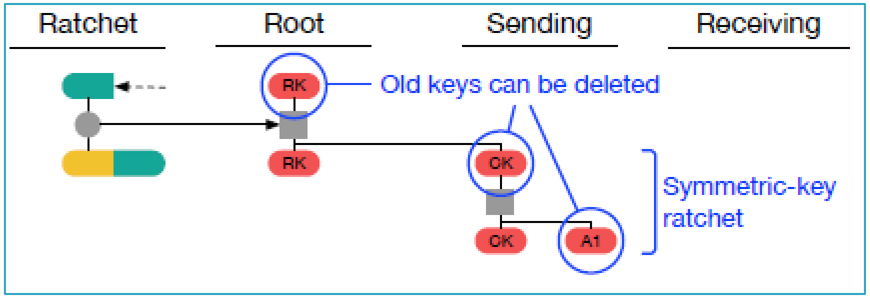


A Double Ratchet Walk-Through

- The following example shows a double ratchet walk-through from Alice’s perspective, including <u>only</u> messages she is receiving from Bob
- Step 1:
  - Alice receives Bob’s public key and generates a new root key (RK) and sending chain key (CK)

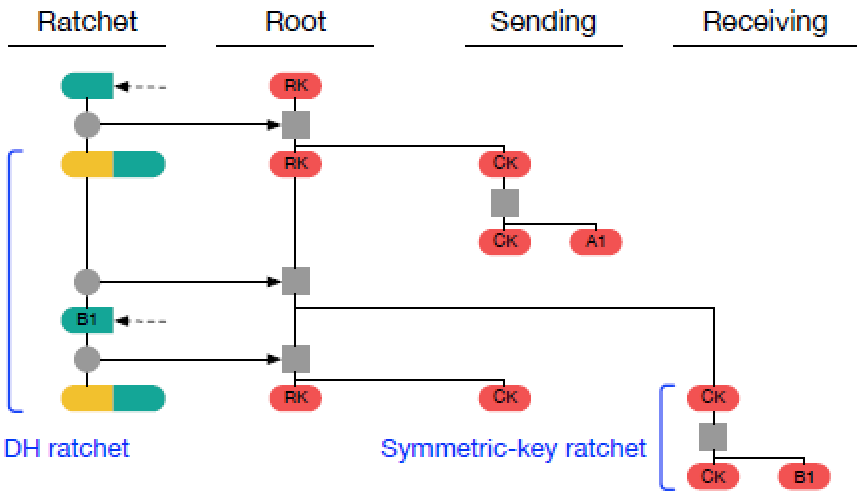


- Step 2:
  - When Alice sends her first message, she applies a symmetric-key ratchet step to her sending chain key (CK), resulting in a
    - message key (A1)
    - new chain key (CK) (ignore for now)



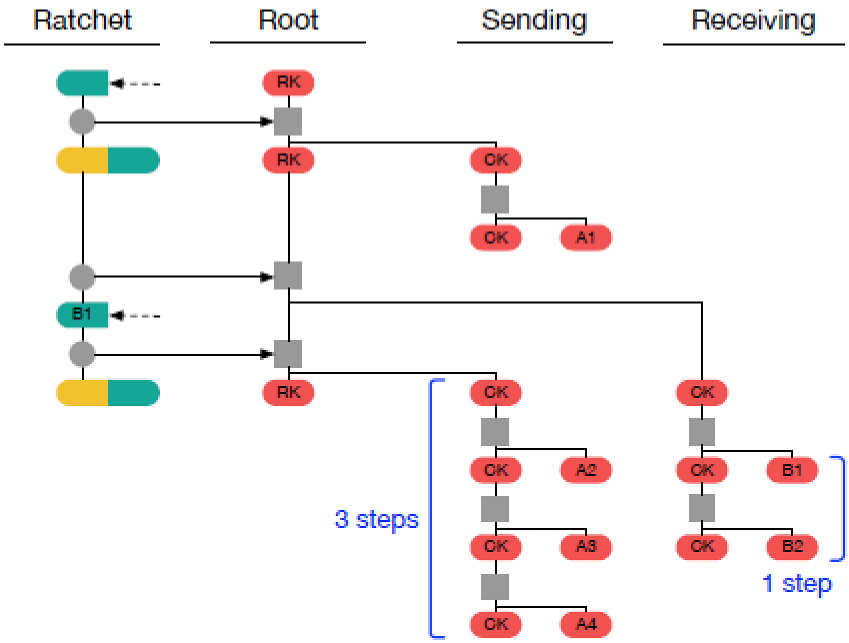
A Double Ratchet Walk-Through

- Step 3:
  - Alice receives a response from Bob; it contains his new DH ratchet public key B1
  - Alice applies a DH ratchet step to derive a new receiving chain key (CK) …
    - She then applies a symmetric-key ratchet step on (CK) to get the message key (B1) for the received message, as well as a new chain key (CK)
  - … and to derive a new sending chain key (CK)
    - In the next step (shown on the next slide), she applies the ratchet on (CK) as well to create the sending key (A2)



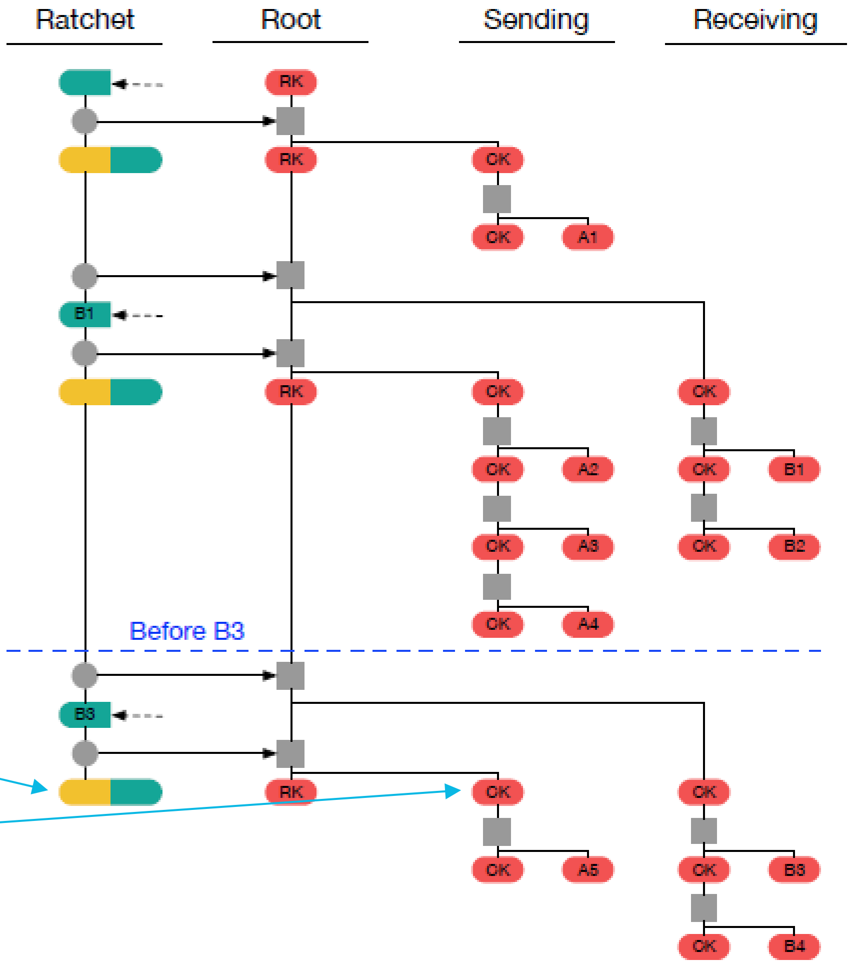
A Double Ratchet Walk-Through

- Step 4:
  - Here Alice next sends a message using (A2), and applies two more ratchet steps to create sending message keys (A3) and (A4) for 2 additional messages
    - Note that the DH-rachet wasn't invoked to create new chain keys, as seen before, i.e. Alice sent a sequence of messages to Bob without prior receiving his new public key
  - Alice receives a message encrypted with (B2)
  - Since Alice didn't receive a new public key from Bob, she simply applies the receiving key ratchet again, to derive (B2)



A Double Ratchet Walk-Through

- Step 5:
  - Alice then receives Bob’s new public key (B3), as well as messages encrypted with (B3) and (B4)
  - She generates these keys, by
    - Applying the DH ratchet and creating a new receiving chain key (CK)
    - Executing the receiving key ratchet twice to generate (B3) and (B4)
  - Alice also generates a new sensing message key (A5), by
    - calculating a new private key
    - Applying the DH ratchet
    - Creating a new sending chain key
    - Executing its ratchet once to create (A5)



Summary: Keys and Key Exchanges in the Double Ratchet Protocol

- Initial Key Exchange:
  - Two parties (Alice and Bob) perform an initial key exchange using (a MitM-resilient variation of) ECDH to establish the *Root key;*
  - The Constants in the symmetric key ratchets are derived from the Root key
- Symmetric Key Ratcheting:
  - Every time a message is sent / received, a new symmetric encryption key is provided by the “send” ratchet and the “receive” ratchet
  - This process is known as "ratcheting forward" and ensures that each message has a unique encryption key
- Asymmetric Key Ratcheting:
  - Normally, after each message exchange, both parties generate a new root key by doing a DH key exchange
  - However, if the message receiver is offline, the sender can still use symmetric key ratcheting to create a new message key for each message

References

[1] The Double Ratchet Algorithm; Trevor Perrin and Moxie Marlinspike

CT437
COMPUTER SECURITY AND FORENSIC COMPUTING

Dr. Michael Schukat

About me

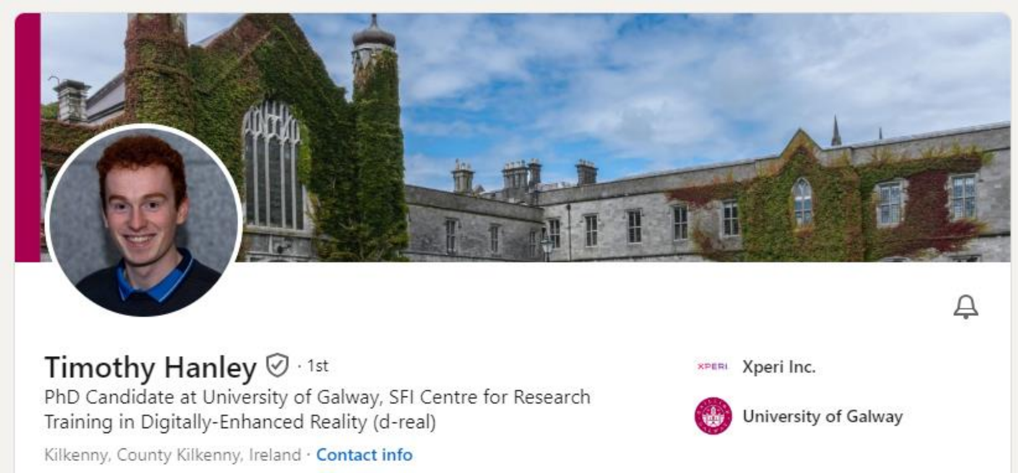
- Professional Background:
  - M.Sc. Computer Science
  - Dr. rer. nat. (Computer Science)
  - (Senior) Lecturer in the School of Computer Science at NUI Galway
  - Senior Embedded Systems Design Engineer (Ireland)
  - Embedded Systems Design Engineer (Germany)
  - Junior Lecturer and Researcher (Germany)
- Research Interests:
  - Many, including cybersecurity
- Contact:
  - michael.schukat@universityofgalway.ie
  - Office CSB3002

My recent Publications in (AI-supported) Cybersecurity

- □ Detecting Ransomware Encryption with File Signatures and Machine Learning Models (2023)
- □ A Security Enhancement of the Precision Time Protocol Using a Trusted Supervisor Node (2022)
- □ The Application of Reinforcement Learning to the FlipIt Security Game (2022)
- □ Precision Time Protocol Attack Strategies and their Resistance to existing Security Extensions (2022)
- □ New Framework for adaptive and agile Honeypots (2020)

Your Lab Tutor

- Timothy Hanley: 2nd year PhD student



Timothy Hanley · 1st

PhD Candidate at University of Galway, SFI Centre for Research Training in Digitally-Enhanced Reality (d-real)

Kilkenny, County Kilkenny, Ireland · Contact info

Cybersecurity versus Computer Security

- **Cybersecurity** is the practice of protecting systems, networks, and programs from digital attacks. These cyberattacks are usually aimed at accessing, changing, or destroying sensitive information; extorting money from users; or interrupting normal business processes Source: Cisco
- **Computer Security** is the historically older term coined at a time when the focus was on individual stand-alone computers rather than entire systems

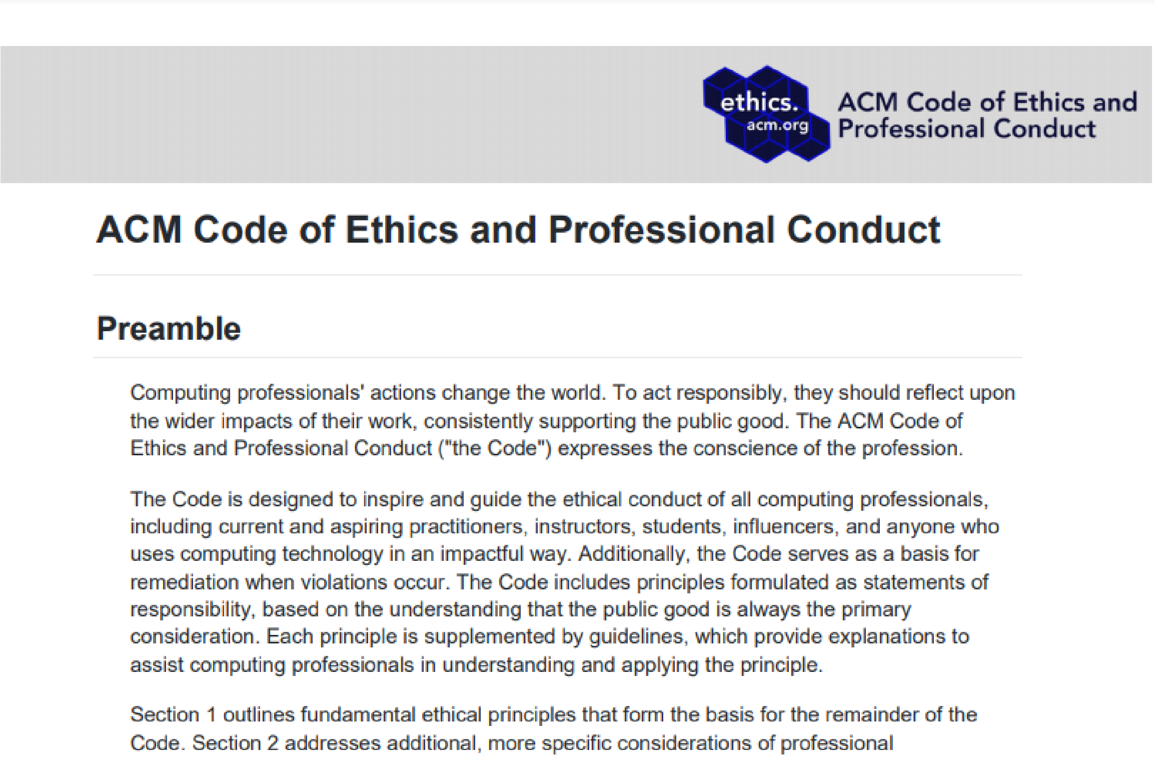
What is Computer Forensics?

- Computer forensics is a branch of digital forensic science pertaining to evidence found in computers and digital storage media
  The goal of computer forensics is to examine digital media in a forensically sound manner with the aim of identifying, preserving, recovering, analysing and presenting facts and opinions about the digital information
  Source: Wikipedia

Some Housekeeping …

Disclaimer

- Please adhere to the **ACM Code of Ethics and Professional Conduct!**
- See Canvas



ACM Code of Ethics and Professional Conduct

Preamble

Computing professionals' actions change the world. To act responsibly, they should reflect upon the wider impacts of their work, consistently supporting the public good. The ACM Code of Ethics and Professional Conduct ("the Code") expresses the conscience of the profession.

The Code is designed to inspire and guide the ethical conduct of all computing professionals, including current and aspiring practitioners, instructors, students, influencers, and anyone who uses computing technology in an impactful way. Additionally, the Code serves as a basis for remediation when violations occur. The Code includes principles formulated as statements of responsibility, based on the understanding that the public good is always the primary consideration. Each principle is supplemented by guidelines, which provide explanations to assist computing professionals in understanding and applying the principle.

Section 1 outlines fundamental ethical principles that form the basis for the remainder of the Code. Section 2 addresses additional, more specific considerations of professional

Use of Canvas

- Announcements
  - **Main communication mechanism, urgent messages may be circulated by email**
- Syllabus
  - Contains module outline, breakdown of marks, etc.
- Modules
  - Compulsory and optional reading materials
- Assessment
- Quizzes
  - In-class quizzes
  - End-of term student feedback questionnaire
- Discussion Forum
  - Mainly used for assignment-related questions
- Quickly attendance (used later for every lecture)
- Virtual Classroom
  - Possibly used for virtual labs

Lecture Organisation / Breakdown of Marks

- 2 hours of lectures per week
  - Wednesday 10:00 – 11:00 in Tyndall Theatre
  - Wednesday 13:00 – 14:00 in ENG-2002
- 2 hours of labs per week (from week 3, tbc)
- There will be a continuous assessment (CA) component worth 30% consisting of
  - 2 assignments
  - in-class quizzes
  - lab worksheets
- The exact CA structure will be shared with you in coming days
- The summer exam has a weight of 70%
  - See Canvas for 2022/23 summer exam
- I'll be also using **Mentimeter** or Vevox for in-class feedback

In-Class Quizzes

- Canvas MCQs, during the lectures
- Open book, addressing content covered during the current or previous week
  - I will provide you with details beforehand
- Typically, 5 randomised questions out of a pool of 20+ questions
- One question is presented at a time, there is no backtracking allowed
- 5 minutes duration

Flipped Learning

- In some lectures we'll apply the concept of **flipped learning:**
  - You'll be notified via Canvas and study the learning materials prior to the weekly lectures
  - If you have specific questions about content, please let me know in good time, so that I can incorporate them into my lecture slots that week

Assignment Content Overview

- The assignments will require you to do the following:
  1. Software development / benchmarking in C using the OpenSSL library
  2. Installation and demonstration of ethical hacking tool (i.e., Metasploit)
     - Extensive use of VM or container
- **Because of various campus restrictions you need to use your own computer / laptop for the assignments**

Some important Ethical Hacking / Penetration Testing Tools

- Kali Linux
  An Advanced Penetration Testing Linux distribution used for Penetration Testing, Ethical Hacking and network security assessments
- Metasploit
  A software platform for developing, testing, and executing exploits
- Shodan
  Shodan is a search engine for Internet-connected devices
  https://www.youtube.com/watch?v=Db5TPYTgy9c

Learning Materials and Textbooks

- Weekly presentations
- There’s no single primary textbook, but William Stalling’s
  - Cryptography and Network Security
  - Data & Computer Communications

  provide a good overview
- I’ll provide you with links to additional sources, e.g.
  - articles
  - eBooks
  - source code

  as we go along

Main Learning Outcomes

On successful completion of this module you will:

1. Have a knowledge of fundamental cybersecurity principles, including confidentiality, integrity, and availability (CIA triad), as well as an understanding of threats and attack techniques by threat actors
2. Have a solid understanding of modern cryptographic algorithms, modern cryptographic network protocols, and their applications
3. Synthesize cryptographic concepts into algorithms / frameworks to address a given cybersecurity problem
4. Be able to conduct simple information / computer system security assessments using ethical hacking / pen-testing strategies and tools
5. Proficient in the use of cryptographic libraries (i.e., OpenSSL)